VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Komprese pomocí neuronové sítě
Data Compression using Neural Networks

2013 Miroslav Čižmárik

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Miroslav Čižmárik**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Komprese pomocí neuronové sítě
Data Compression using Neural Networks

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat algoritmus pro kompresi dat pomocí neuronové sítě. Navržený algoritmus může být převzat z literatury.

Obsahem práce bude:
1. Popis neuronových sítí.
2. Popis algoritmu pro kompresi pomocí neuronové sítě.
3. Implementace algoritmu.
4. Porovnání efektivity algoritmu s jinými kompresními metodami.

Seznam doporučené odborné literatury:

[1] Data Compression: The Complete Reference, David Salomon, 4. ed., Springer, 2007
[2] M. Mahoney, Fast Text Compression with Neural Networks, Proc. AAAI FLAIRS, Orlando, 2000 (C) 2000, AAAI.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Platoš, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

PROHLÁŠENÍ

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Miroslav Čižmárik

21.7.2013

PODĚKOVÁNÍ

Ďakujem svojmu vedúcemu práce Ing Janovi Platošovi, Ph.D. za podporu, trpezlivosť a najmä poskytnutie cenných rád, ktoré mi pomohli pri štúdiu a spracovaní mojej bakalárskej práce.

ABSTRAKT

Tato bakalářská práce se zabývá kompresí dat pomocí neuronových sítí a srovnává výhody tohto řešení oproti jiným kompresním metodám. Prezentuje, co jsou to neuronové sítě a na jakých principech jsou založeny. Zkoumá souvislosti mezi kompresí dat a neuronovou sítí. Předkládá také základní myšlenky jiných principů učení neuronových sítí a odlišných kompresních metod bez detailního popisu. Představuje pravděpodobnost odhadu bitové následnosti při zmenšování objemu dat. Jejím hlavním cílem je vypočítání entropie velikosti souboru po komprimaci se zaměřením na textové a bitmapové formáty.

ABSTRACT

This work deals with data compression using neural networks and compares the advantages of this solution compared to other compression methods. Presents, what the neural networks are and the principles on which they are based. It examines the relation between data compression and neural networks. It is also presenting a basic idea of the other principles of learning neural networks and different compression methods without detailed description. Represents the probability estimate of the bit sequence in reducing the amount of data. Its main objective is to calculate the entropy file size after compression with a focus on text and bitmap formats.

KLÍČOVÁ SLOVA

Neuronová síť, neuron, komprese, entropie, perceptron

KEYWORDS

Neural network, neuron, compression, entropy, perceptron

ZOZNAM POUŽITÝCH SKRATIEK

ART – Adaptive Ressonance Theory

Dôvera – váha na synapsii medzi dvomi neurónmi

E – chyba

NN – neurónová sieť (neural network)

PPM – prediction by partial match

RE – rekurentné siete

SV – synaptické váhy

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

Obsah

ÚVOD

V tejto práci sú predstavené neurónové siete, ich vznik, kategorizácia a použitie.

Všetky umelé neurónové siete a ich metódy učenia sa sú odpozorované z biologických sietí v živých organizmoch. Vlastnosti týchto sietí sa rozvíjali súbežne s novými objavmi v biológii, čo dokazujú aj spôsoby trénovania týchto sietí.

Umelé neurónové siete sa vyvíjali od najjednoduchšej siete vo forme perceptrónu až po viacvrstvové siete z ktorých trojvrstvé sú jedny z najrozšírenejších v súčasnom vývoji aplikácií. Neoddeliteľnou súčasťou poznatkov o neurónoch je aj aktivačná funkcia zaoberajúca sa vstupmi do neurónov. Modifikuje synaptické váhy a tým umožňuje vykonávať samotný proces učenia sa.

Učenie, ako neoddeliteľná súčasť poznávania, sa aplikuje v umelých sieťach pri procese rozpoznávania a zaraďovania do jednotlivých skupín.

Na rôzne typy problémov sa používahú rozličné typy sietí a odlišné metódy ich učenia sa. Medzi často používané patria metódy ako online learning, či error backpropagation. Zaujímavú ukážku tvoria Hopfieldove siete, ktoré sú výrazne odlišné od dopredných sietí, pretože sa viac podobajú ľudskému prepojeniu neurónov.

Nutnou zložkou mojej práce je kompresia dát. Predstavujem dva spôsoby kompresie – pomocou Huffmanovho kódu a aritmetické kódovanie. Na tieto dva druhy kompresie som sa zameral z dôvodu ich častej používateľnosti.

V závere mojej práce zhrňujem dosiahnuté výsledky a zároveň navrhujem iný spôsob kompresie vychádzajúci z myšlienky zapisovania údaju o najbližšej chybnej predpovedi. Práca môže poslúžiť v rýchlejšej a kvalitnejšej kompresii z dôvodu možnosti rozdelenia celého výpočtu do vlákien. Poukazuje na nový štýl myslenia, uchovávania a navrhovania zápisu dát so zatiaľ neprebádanými možnosťami bezstratovej kompresie.

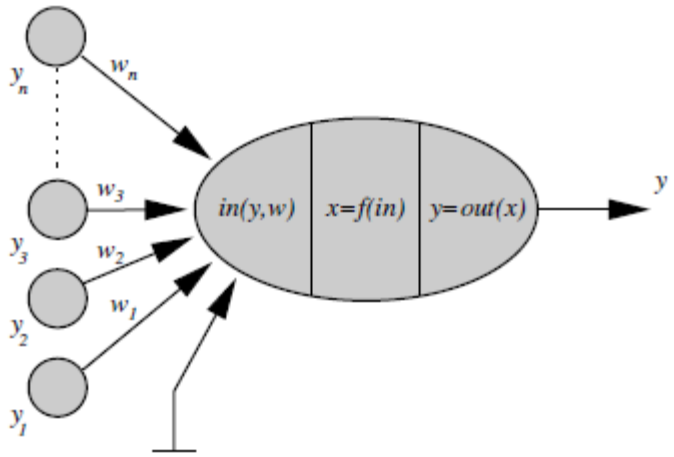
1. NEURÓNOVÁ SIEŤ

1.1 NEURÓN

Neurón ako ho popisujú *McCullochom* a *Pittsom* je základná jednotka neurónovej siete, ktorá má excitačné (zosilňujúce) a inhibičné (zoslabujúce) vstupy. Stavom ich neurónu je vždy hodnota reprezentujúca hodnotu 0 alebo 1. Tomuto neurónu hovoríme *logický neurón. Logický neurón* je schopný simulovať len lineárne separovateľné Boolove funkcie. Inak povedané, nie je schopný simulovať funkcie ako je XOR.

Základnými vlastnosťami neurónu sú:

1. prijímanie signálov z okolia
2. spracovanie prijatého signálu
3. posielanie spracovaných informácií iným neurónom z jeho okolia



Obrázok 1: *Štruktúra neurónu* [2]

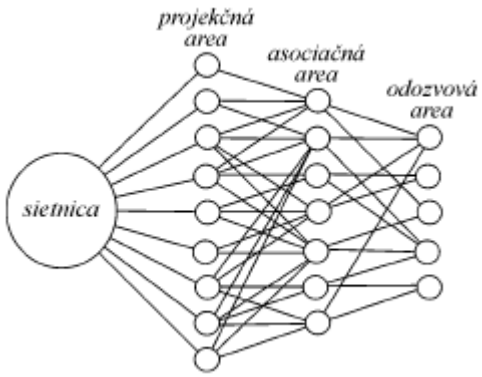
Časti z ktorých neurón pozostáva:

1. vstup do neurónu (dendrid)
2. prah neurónu
3. aktivačná funkcia neurónu
4. výstupná funkcia neurónu
5. synaptické váhy (váhy na prepojeniach medzi neurónmi)

Nevýhodou *neurónu* bola neschopnosť učenia sa. Tento nedostatok však vyriešil ***Frank Rosemblatt***, keď predstavil svoj *perceptrón*.

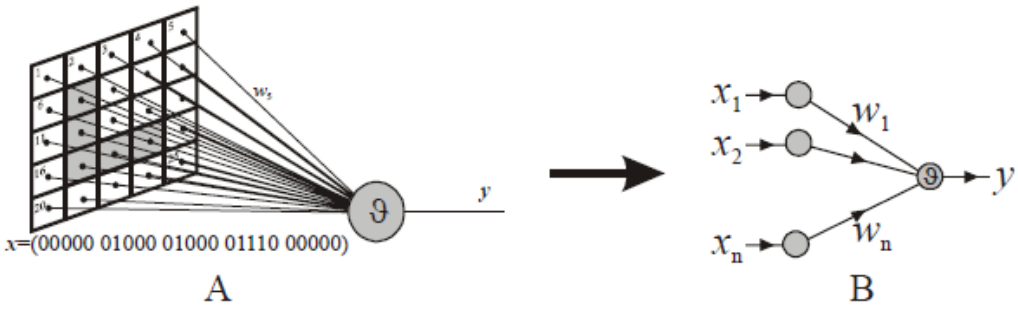
1.2 PERCEPTRÓN

Zásluhou *Franka Rosemblatta* bolo učenie zahrnuté do konštrukcie neurónu, pričom váhové a prahové koeficienty sa začali považovať za premenné danej konštrukcie neurónu. Tieto premenné sa nastavujú procesom učenia sa.



Obrázok 2: *Rosemblattova predstava o perceptróne* [8]

Rosemblattova myšlienka o *perceptróne* bola naučiť sa správne identifikovať objekt z obrazu na sietnici tak, aby táto identifikácia v odozvovej oblasti bola správne rozpoznaná a zaradená. Pozorovaný objekt sa premieta na sietnicu, ktorá prenáša binárne údaje do projekčnej oblasti (arei). Tu sa tieto údaje predspracuvávajú. Pričom rozlišovanie je len dichotomické, teda sa len zisťuje, či daný vstup patrí do rozpoznávanej skupiny, alebo patrí do skupiny ostatných. Majme na pamäti, že vstupy (spoje) zo sietnice do projekčnej oblasti sú nemenné. Teda vstupy sa meniť nikdy nebudú, vďaka čomu im v kapitole o implementovaní neurónových sietí nebudeme nastavovať prahy (5.2 prahy sa budú nastavovať až na synapsiách nie na vstupe). Ostatné prepojenia medzi projekčnou, asociačnou a odozvovou oblasťou budú menené v závislosti na naučených poznatkoch. Jednotlivými vrstvami sa viac zaoberám v kapitole 1.7.



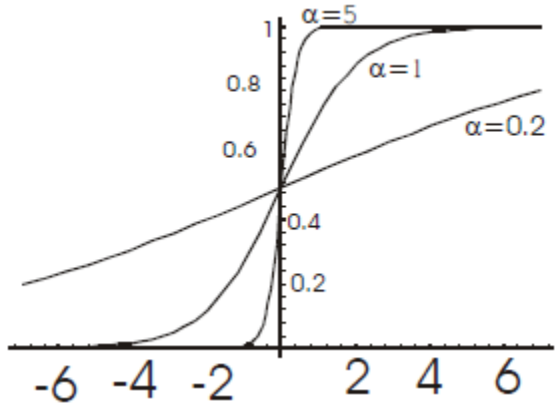
Obrázok 3: *Minského a Papertova predstava o perceptróne* [8]

Minského a Papertova predstava o perceptróne popisuje dve vrstvy. Ich zámerom je to, že perceptrón nebude rozlišovať a rozraďovať všetky podnety. Naopak, bude reagovať práve na jeden takýto podnet. Tento druh perceptrónu je najzákladnejšou dvojvrstvou neurónovou sieťou, lebo všetky spoje medzi vstupmi a výstupmi majú váhové koeficienty w_i na nastavovanie dôležitosti daného vstupu a výstupný neurón má prah ϑ .

1.3 AKTIVAČNÁ (PRENOSOVÁ) FUNKCIA

Je to funkcia neurónového vstupu. Teda funkcia, podľa ktorej sa modifikuje vstup do neurónu.

Aktivačné funkcie môžu byť rôzne. Medzi najpoužívanejšie sa radia funkcie ako je *lineárna funkcia* ($x_i = f(in_i) = in_i$), *funkcia signum* ($x_i = f(in_i) = 1$, *ak* $in_i \geq 0$; $x_i = f(in_i) = 0$, *ak* $in_i < 0$), *po častiach lineárna funkcia* ($x_i = f(in_i) = 1$, *ak* $in_i \geq 1/2$; $x_i = f(in_i) = 0$, *ak* $in_i \leq -1/2$; $x_i = f(in_i) = in_i$, *ak* $in_i \leq 1/2 \cap in_i \leq -1/2$), *sigmoidálna funkcia* ($x_i = f(in_i) = 1/(1+e^{-\alpha in_i})$). Pričom najviac používanou je *sigmoidálna funkcia* zobrazená na obrázku dole (Obrázok 4).



Obrázok 4: *Sigmoidálna funkcia* [8]

Ako už názov napovedá, *neurónová sieť* je tvorená neurónmi, pričom má schopnosť učiť sa. Jej životný cyklus má dve fázy. Fázu učenia sa a fázu života siete. Najviac momentálne preferovanou neurónovoou sieťou je trojvrstvá neurónová sieť (*three-layer network*).

Voľne preložené z definície v diele *Theory of the Backpropagation Neural Network* [14]. Neurónová sieť je paralelná štruktúra, ktorá distributívne spracováva informácie pomocou spracovávateľských prvkov, z ktorých sa skladá (prvky môžu obsahovať lokálnu pamäť a môžu vykovávať lokálne operácie spracovania informácií). Tieto prvky sú navzájom poprepájané jednosmernými signálovými kanálmi nazývanými spojenia (connections). Každý spracovávateľský prvok má jediné výstupné prepojenie, ktoré vybočuje „z ohrady“ do toľkých vedľajších spojení, koľko požadujeme. (Každé nesie rovnaký signál – výstupný signál spracovaného prvku.) Výstupný signál spracovávateľského prvku môže byť akéhokoľvek požadovaného matematického typu. Každé spracovávanie prebiehajúce v spracovávanom prvku musí byť úplne lokálne. Teda musí závisieť len

na aktuálnych hodnotách vstupného signálu, prichádzajúceho do spracovávateľského prvku pomocou kontaktného spojenia a na hodnotách uložených v lokálnej pamäti spracovávateľského prvku.

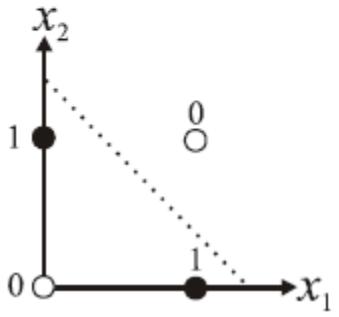
Neurónová sieť je schopná funkciu ako je XOR realizovať, aj keď táto funkcia nie je lineárne separovateľná. Dôvodom tejto možnosti je jej realizácia pomocou *logického neurónu vyššieho rádu* (konkrétne druhého rádu).

Životný cyklus neurónovej siete sa skladá z dvoch etáp. Prvou etapou je proces učenia sa, kedy sa snažíme sieť naučiť závislostiam pomocou trénovacej množiny a metódy na zapamätanie si informácií nastavením synaptických váh. Samozrejme, k tejto etape patrí aj testovanie danej siete pomocou testovacej množiny v prípade, že sú nám známe výsledky danej siete (existujú prípady, kedy výsledky nemáme k dispozícii). Druhou etapou je už samotný život siete, ktorý pozostáva z vykonávania činnosti, na ktorú bola daná sieť pripravovaná.

1.4 VIACVRSTVÉ NEURÓNOVÉ SIETE

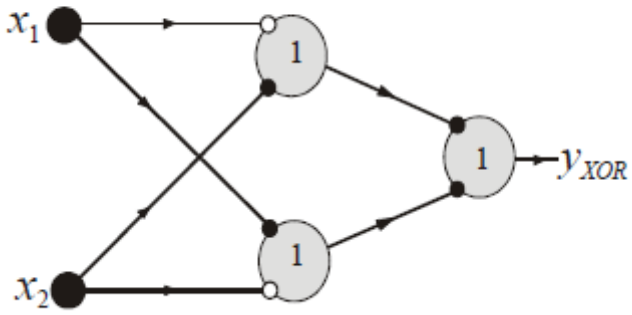
Keďže aj v živote si rozkladáme riešené úlohy na podúlohy, tak aj v neurónových sietiach sa snažíme dosiahnuť takýto rozklad. Zatrieďovacie a iné rozhodnutia rozdelíme na menšie jednoduchšie rozhodnutia, z ktorých dôjdeme k jednotnému požadovanému prvotnému výsledku. To je hlavnou myšlienkou vzniku viacvrstvých neurónových sietí. Ukážkou správnosti môže byť napríklad aj funkcia XOR, ktorá nie je lineárne separovateľná (Obrázok 5) a základný neurón si s ňou neporadí. Avšak pri použití trojvrstvej siete tento problém vyriešime vďaka rozložiteľnosti XOR funkcie na čiastočné už lineárne separovateľné funkcie (Obrázok 6). To je tiež príkladom vety *Minského a Paperta* (v knihe *Perceptron*) :“ *Ľubovoľná Boolova funkcia f je simulovaná logickým neurónom vyššieho rádu.“* (Ing. Michal Čerňanský, PhD.; materiály k prednáškam z predmetu Neurónové siete, slajdy 2 priesvitka 30 [8]).

#	x	y	$\varphi_{XOR}(x,y)$
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0



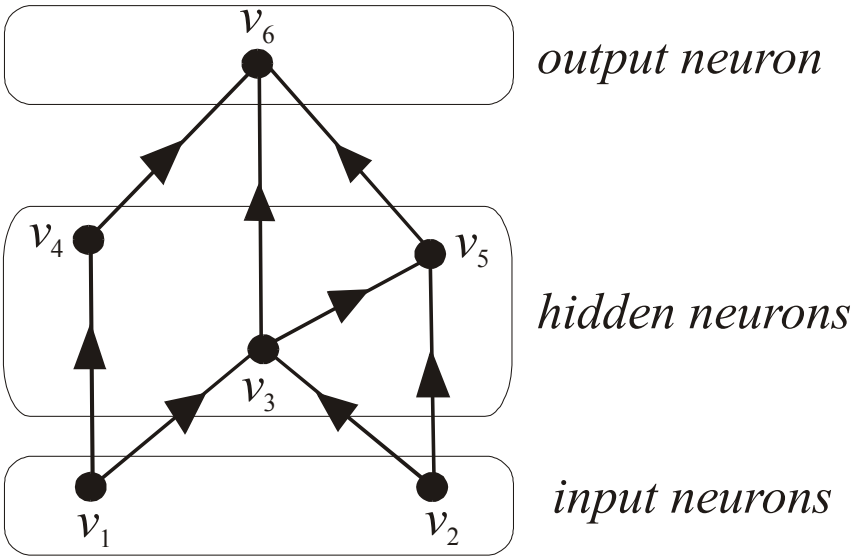
Obrázok 5: *Neseparovateľnosť funkcie XOR* (na obrázku je vidieť, že nie je možné niako rozdeliť jednou priamkou priestor na dva podpriestory) [8]

$$\varphi_{XOR}(x_1, x_2) = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$$



Obrázok 6: *Ukážka možnosti skonštruovania funkcie XOR pomocou trojvrstvej neurónovej siete* [8]

Viacvrstvové neurónové siete sa vyznačujú tým, že majú aspoň tri a viac vrstiev neurónov. Za zakládné tri vrstvy sa považujú postupne od najnižšej *vstupná (input layer), skrytá (hidden layer) a výstupná (output layer) vrstva*. Všetky vrstvy neurónových sietí musia byť navzájom úplne poprepájané. To znamená, že každý neurón z nižšej vrstvy musí byť spojený s neurónom z vyššej vrstvy. (Obrázok 7)



Obrázok 7: *Prepojenie neurónov na trojvrstvej neurónovej sieti* [8]

Postupom spracovávania informačných dát v týchto sietiach je podľa diela *Neuronové sítě* (prof. Ing. Ivo Vondrák, CSc., 2009 [11]) takýto:

1. Najskôr sú excitované na odpovedajúcu úroveň (v rozmedzí 0 až 1) neuróny vstupnej vrstvy.
2. Tieto excitácie sú pomocou väzieb privedené k nasledujúcej vrstve a upravené (zosilnené, či zoslabené) pomocou synaptických váh.
3. Každý neurón tejto vyššej vrstvy spraví sčítanie upravených signálov od neurónov nižšej vrstvy a je excitovaný na úroveň danú svojou aktivačnou funkciou.
4. Tento proces prebieha cez všetky vnútorné vrstvy až k vrstve výstupnej, kde potom získame excitačné stavy všetkých jej neurónov.

1.5 UČENIE SA

Neurónové siete sa môžu učiť pod dohľadom alebo bez dohľadu.

1.5.1 POD DOHĽADOM (KONTROLOVANÉ UČENIE)

Vyznačuje sa prítomnosťou učiteľa počas celej doby učenia sa. Kontrolované učenie môžeme ešte rozdeliť na *štrukturálne učenie* a *temporálne učenie.*

Štrukturálne učenie môže byť *autoasociačné.* Vtedy na vstup neurónovej siete dávame rovnakú vzorku ako na jej výstup, aby sme dosiahli adaptáciu neurónovej siete. Tento druh sietí sa používa napríklad na simuláciu pamäte.

Heteroasociačná sieť roztrieďuje vzorky do skupín tak, že vstupné vzorky zatrieďuje podľa učiteľa.

Temporálne učenie, oproti *štrukturálnemu,* je založené na sekvencii vstupov v čase. Až potom je sieti priradený jeden výstup (napríklad šachová partia).

Učenie môžeme rozdeliť aj podľa prístupu k zmene synaptických váh - a to na učenie založené na *oprave chyby (error correction learning)* a *stochastické učenie (stochastic learning)*. Pričom existuje aj *posilnené učenie (reinforcement learning)*, ale tento druh nie je vo všetkých publikáciách zaradený pod učenie s učiteľom, pretože má aj prvky učenia bez učiteľa.

V učení pomocou *opravy chyby* sa váhy na synapsiách upravujú pomocou vypočítania chyby medzi očakávaným výsledkom a výsledkom skutočným.

Pri *stochastickom učení* používame princíp postupných krokov. Teda ako sa píše v diele *Neurónové siete Inžiniersky prístup* (1996) [2]: „

- *navrhne sa stochastická zmena SV a vypočíta sa energia NN*
- *ak zmena priniesla zníženie energie NN, návrh zmeny sa prijme*
- *ak zmena nepriniesla spomínaný efekt, návrh sa zamietne*“

(posilnené učenie je vysvetlené v stati 1.5.3)

Metódu *pod dohľadom* aplikujeme tak, že si pripravíme mnoho testovacích príkladov, pričom pri týchto príkladoch musíme mať presne špecifikované ich vstupy a očakávané hodnoty výstupov. Potom časť príkladov postupne vyskúšame a po odskúšaní každého jedného príkladu z podskpupiny príkladov skúsime upraviť váhy spojujúce neuróny tak, aby veľkosť novovzniknutej chyby bola nižšia ako predtým. Teda spravíme jednu epochu. Ak je veľkosť epochy nižšia ako celý tréningový set, tak je dôležité, aby podskupina príkladov bola vždy vyberaná náhodne, inak môže dôjsť k nežiadúcim osciláciám. [1]

1.5.2 BEZ DOHĽADU (NEKONTROLOVANÉ UČENIE)

Pri tomto prístupe tiež máme skupinu príkladov, avšak nemáme výstupy. Predpokladáme, že výstupy patria do určitých skupín a neurónová sieť priradí daný výstup určitej skupine. Tréning spočíva v nechaní siete, aby zistila najvýznamnejšie rysy a použila ich k priraďovaniu k jednotlivým skupinám.

Týmto sieťam sa ponúkne iba vstup, ktorý sieť sama spracuje, a určí výstup na základe daných zákonitostí. Ukončenie učenia nastáva, keď je váha v danom a v nasledujúcom kroku menšia ako požadovaná odchýlka ε. Ako príklady môžeme uviesť *Hebbovo učenie (Hebbian learning*) a *Kooperačné a konkurenčné učenie (Cooperative and competitive learning)*.

1.5.3 POSILNENÉ UČENIE (REINFORCEMENT LEARNING)

Je hybridnou metódou kombinujúcou predchádzajúce dve metódy.

Bez dohľadu je v tom zmysle, že nepoznáme výsledky, ale zároveň pri odpovedi na príklad z tréningového setu je sieti oznámené, či bola daná odpoveď správna, alebo nie.

1.6 HEBBOVO PRAVIDLO

Toto pravidlo špecifikuje o koľko sa má spojenie medzi dvomi vrstvami zmeniť vzhľadom k ich aktivácii. Teda keď sa neurón A opakovane zúčastňuje na aktivácii neurónu B, tak sila akcie z A na B rastie ako sa píše v [10] (Cornell University). Hebb túto myšlienku predstavil vo svojej knihe "Organizácia správania" (*The Organization of Behavior, 1949*)

Všeobecnejším pravidlom Hebbovho učenia je:

$$\Delta w_{ij}(t) = F\big(x_j, x_i, \lambda, t, \theta\big),$$

kde na tejto jednej synapsii x_j je výstupom vstupného neurónu (prvého na začiatku synapsie), x_i je výstupom výstupného neurónu(druhého na konci synapsie), w_{ij} je sila ich spojenia a λ je learning rate (smernica učenia sa – ako rýchlo sa sieť adaptuje). Sila tohto spojenia stúpa pokiaľ oba neuróny sú aktivované „simultálne“ [7]. „Simultálne“ je však v tomto prípade veľmi závislé od danej siete a môže to byť niekoľko milisekúnd, ale aj niekoľko hodín (napríklad učenie sa závislostí u psa v porovnaní s učením sa závislostí u človeka). Opačne, ak sa neuróny na opačných stranách synapsie aktivujú v rôznych časoch, tak sa váha ich spojenia znižuje.

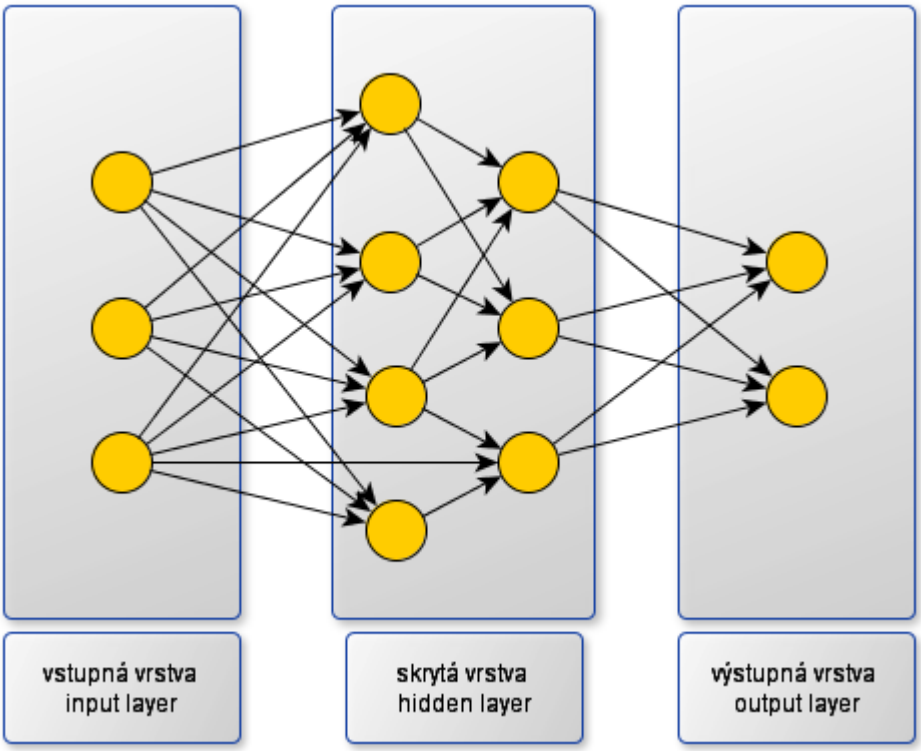
1.7 DOPREDNÉ NEURÓNOVÉ SIETE

Dopredné (feed-forward) siete sú tie, v ktorých sa informačné dáta (signál) šíria len jedným smerom. V týchto sietiach nie je problém rozdeliť jednotlivé neuróny do troch základných skupín.

Prvou skupinou (vrstvou, oblasťou) by boli vstupné neuróny, ktoré sa vyznačujú tým, že komunikujú s okolitým svetom, z ktorého dostávajú vstupy, a s neurónmi ktorým ďalej posielajú spracované dáta. Túto vrstvu nazývame *vstupnou (input layer)*.

Druhú skupinu (vrstvu, oblasť) nazývame ako *skrytú vrstvu (hidden layer)*. Táto vrstva obvykle zahŕňa viac „podvrstiev“, pretože všetky neuróny v nej obsiahnuté posielajú dáta ďalším neurónom a tie ich môžu posielať ďalším neurónom. Teda aj tieto ďalšie neuróny patria do tejto vrstvy. Ako som už opísal v tejto vrstve sa nachádzajú neuróny, ktoré komunikujú iba s neurónmi a to ako na vstupe, tak aj na ich výstupe.

Tretia skupina (vrstva, oblasť) je *výstupná (output layer),* v ktorej neuróny prijímajú informačné dáta z neurónov a na výstupe komunikujú s externým svetom.



Obrázok 8: *Vrstvy doprednej neurónovej siete*

1.8 DÁVKOVÉ UČENIE (BATCH LEARNING, STATISTICAL LEARNING)

Ako pri online učení (1.9) aj pri dávkovej metóde učenia sa snažíme vyriešiť problém typu pridelenia výstupných výsledkov (množina B) vstupom (množina A) [5]. V dávkovom učení predpokladáme, že existuje pravdepodobnostné rozloženie na celej množine kartézskeho súčinu vstupov a výstupov (A × B) a že máme prístup k časti tejto množiny v podobe tréningových setov. Algoritmus dávkového učenia používa tréningový set na generovanie výstupných hypotéz, ktoré sú funkciami mapujúcimi vstupné inštancie na výstupy. Očakávame, že tento algoritmus bude generalizovať v tom zmysle, že použije naučené prípady z predložených tréningových setov na ešte neodskúšané a neznáme prípady ktoré sa bude snažiť rozdeliť na základe už naučených podobností a pravdepodobností. Všetky prípady, na ktoré chceme danú metódu použiť, musia pochádzať z množiny A × B.

1.9 ONLINE LEARNING (SEQUENTIAL, PATTERN-BASED)

Online learning, podobne ako métoda batch learning, sa snaží každému prvku vstupnej množiny priradiť prvok z výstupnej množiny, pričom obe tieto množiny sú vopred dané [15].

V metóde online learning zvyčajne nerobíme žiadne štatistické predpoklady týkajúce sa pôvodu dát. Algoritmus v tomto prípade dostane sekvenciu tréningových príkladov, ktoré postupne po jednom spracováva. V každom kole, pre konkrétnu jednu inštanciu, predpovie hypotézu, do akej skupiny pravdepodobne daný vstup (inštancia) patrí a túto predpoveď si zapamätá. Po tejto predpovedi algoritmus obdrží správnu odpoveď do ktorej skupiny daný vstup skutočne patril. Túto odpoveď použije na úpravu vlastných parametrov (synaptických váh), aby sa zlepšila jeho predpovedacia schopnosť. Nie je tu žiadny náznak štatistického generalizovania, pretože iba očakávame, že algoritmus bude presnejšie predpovedať skupiny do ktorých vstupy patria.

Tak ako som už spomínal, algoritmus sa vyznačuje okamžitým aktualizovaním váh po každom vloženom tréningovom sete. Aktualizovaním váh sa pritom nemyslí trénovanie počas normálnej doprednej operácie siete (zahŕňa „off-line“ tréning rovnaký ako pri batch móde). Ich funkcia je založená na týchto vzorcoch:

$$w_i^{(t+1)} = w_i^t - \lambda \frac{\partial E_k(w^{(t)}, \vartheta^{(t)})}{\partial w_i}, alebo\ w_i^{(t+1)} = w_i^t + \lambda(\hat{y}_k - y_k)t'(\xi_k)x_{ik},$$

$$\vartheta^{(t+1)} = \vartheta^{(t)} - \lambda \frac{\partial E_k(w^{(t)}, \vartheta^{(t)})}{\partial \vartheta}, alebo\ \vartheta^{(t+1)} = \vartheta^{(t)} + \lambda(\hat{y}_k - y_k)t'(\xi_k),$$

kde w_i sú synaptické váhy v čase (cykle), λ je parameter rýchlosti učenia sa a ϑ je prah.

Online učenie nezahŕňa skutočnú metódu postupného znižovania gradientu, pretože suma derivácií v danom iterátore nie je nikdy určená pre neaký určitý set váh. Váhy sú namiesto toho ľahko pozmenené až po každom sete (vzore). Online model nie je jednoduchou aproximáciou metódy postupného znižovania gradientu, nakoľko jednotlivé derivácie, ako skupina sčítancov gradientu, majú náhodné odchýlky, ktoré nemusia byť malé. Aj keď sa chyba väčšinou zmenšuje po viacerých zmenách na váhach, môžu sa vyskytnúť deriváty, ktoré naopak budú zvyšovať aj chybu. Pokiaľ rýchlosť učenia nie je veľmi malá, tak váhový vektor má tendenciu vychyľovať sa nad hladinu E(w). Väčšinou má zostupujúci charakter, ale niekedy môže prudko vzrastať. Veľkosť týchto výkyvov je proporcionálne daná rýchlosťou učenia sa (epsilonu).

Pri tomto druhu tréningu sa vyvarujeme používaniu cyklicky sa opakujúcich tréningových setov, pretože by to mohlo limitovať konvergenciu váh len na daný cyklus. Taktiež sa pri tomto druhu tréningu odporúča pri veľkom počte tréningových príkladov používať náhodné prechádzanie príkladov, aby potenciálne nevznikla chyba spôsobujúca nastavovanie synaptických váh v odlišnom smere, než sme predpokladali v prípade usporiadaného prechádzania tréningových príkladov.

Výhody online prístupu oproti batch prístupu je napríklad v rýchlejšom prispôsobovaní váh pri veľkých dátach s vysokou redundanciou ako aj jednoduchosť implementácie a výpočtová efektívnosť.

1.10 METÓDA SPÄTNÉHO ŠÍRENIA CHYBY (ERROR BACKPROPAGATION)

Najväčším predstaviteľom tejto metódy bol Rumelhart a skupina PDP, ktorí túto metódu zviditeľnili medzi verejnosťou a dali jej použiteľnú podobu. Obľúbenosť si získala vďaka schopnosti naučenia sa komplikovaných multidimenzionálnych mapovaní [13].

Táto metóda funguje na princípe propagácie informácie z vyšších vrstiev do nižších. Teda každý neurón v riadku z nižšej skrytej vrstvy získa spätnú chybovú korekciu od neurónu z vyšieho riadku. Inak povedané, každé jedno spojenie nesie iný signál spätnej korekcie.

Funkcia sa vykonáva v dvoch fázach - v doprednom a v spätnom poslaní signálu. To znamená, že v jednom cykle sa najskôr aplikuje prenosová funkcia v doprednej fáze a spočíta sa výsledok, potom nastane spätná fáza, ktorá prenastaví synaptické váhy na spojoch (synapsiách) podľa vzniknutej chyby a proces sa opakuje, kým nenastane ukončovacia podmienka. Je dôležité si uvedomiť, že hodnota nastavená v pribehu aktualizácie sa nemení do ďalšej fázy, kedy sa znova prestaví pri spätnej propagácii.

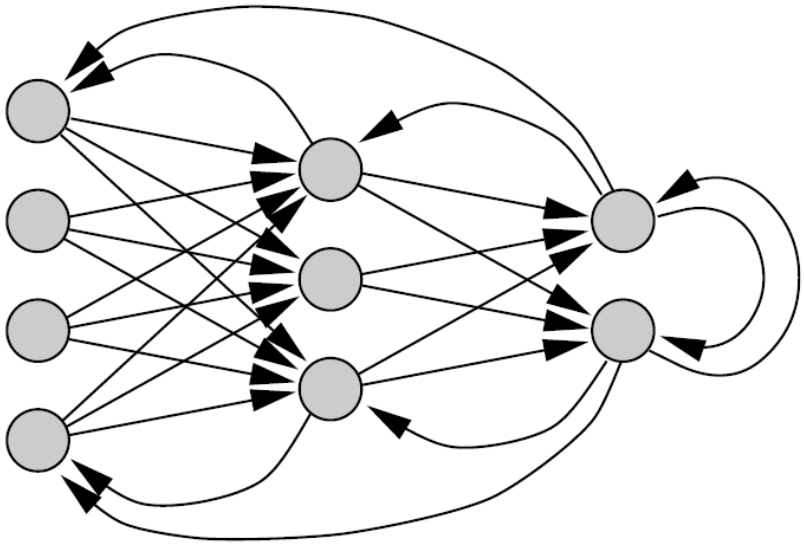
Na zmeranie presnosti aproximácie funkcie a zistenie výslednej chyby použijeme vzorec:

$$F(w) \equiv \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} F_k,$$

kde *F(w)* predstavuje celkovú priemernú chybu, pričom *F(w)* je nezáporné, lebo sčítavame nezáporné čísla. Keďže v našom prípade sa blížime počtom vzoriek k nekonečnu, tak podľa *Kolmogorového pravdepodobnostného teorému* náhodná premenná F(w) sa riadi silným pravidlom veľkých čísel. Teda pri veľkom množstve vzorkov je zanedbateľné, či niekoľko málo z nich je nesprávnych a zavádzajúcich, lebo ostatné vzorky vynahradia tento deficit. V našom prípade nebudú mať dopad na celkovú chybu - respektíve budú mať zanedbateľne malý dopad. To znamená, že naša suma *F* musí skoro isto konvergovať k očakávanej hodnote *F(w)* z *F(k)*.

2 REKURENTNÉ NEURÓNOVÉ SIETE

V týchto sietiach je obtiažne rozdeliť neuróny do skupín ako pri dopredných sietiach, pretože neuróny často zastávajú úlohu vstupných, ale aj výstupných neurónov. Rozumieme tým, že majú synapsie orientované rôznymi smermi (*duálny neurón*).

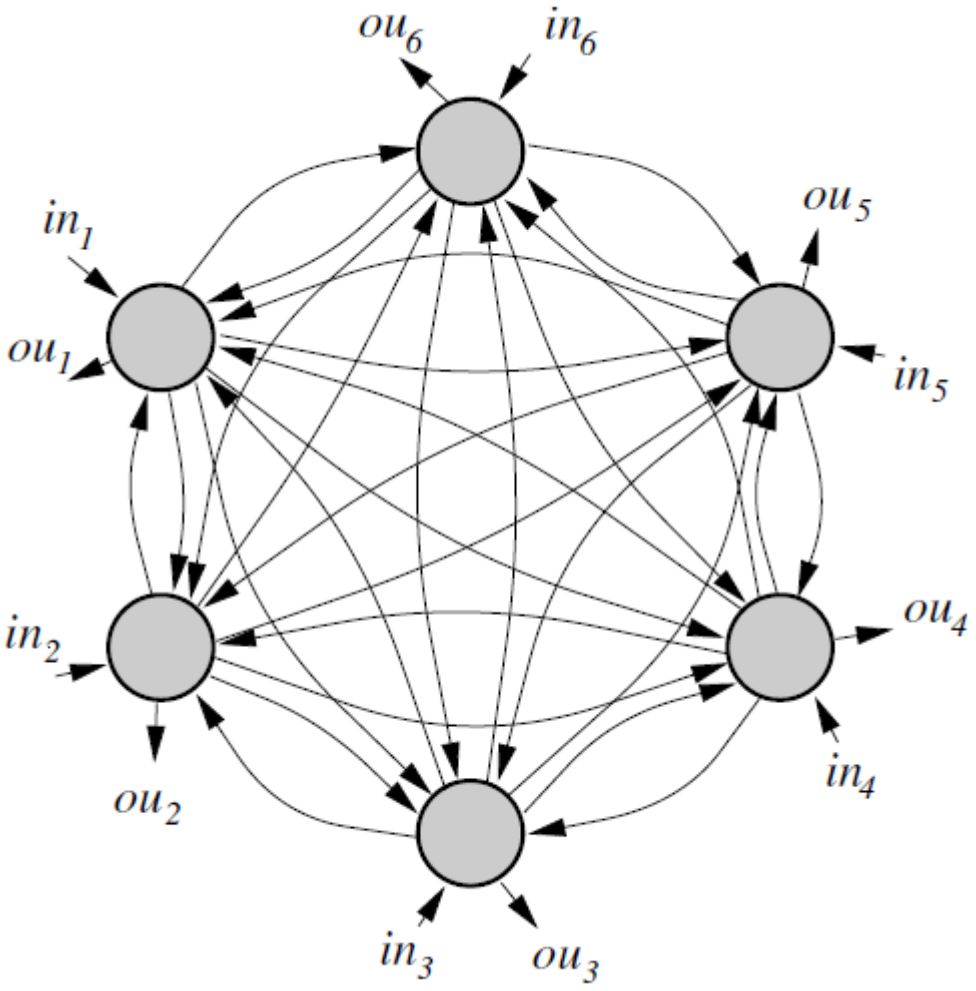


Obrázok 9: *Rekurentná neurónová sieť* [3]

Tieto siete sa taktiež delia podľa učenia na *kontrolované* a *nekontrolované (učenie)*. Ďalšou podskupinou by mohli byť čiastočne rekurentné siete, v ktorých existujú neuróny, ktoré nespĺňajú vlastnosti rekurencie.

2.1 KONTROLOVANÉ UČENIE

2.1.1 HOPFIELDOVE SIETE



Obrázok 10: *Jednoduchá Hopfieldova sieť* [3]

Sú najjednoduchším predstaviteľom rekurentných neurónových sietí. Skladájú sa z navzájom poprepájaných neurónov (Obrázok 10), ktoré menia svoje stavy asynchrónne. Dôležité je všimnúť si, že vstupom do neurónov je ako vstup z ostatných neurónov (synaptické spoje), tak aj vstup z okolitého sveta. $in_i(t) = \sum_{j=1\& j\neq i}^{N} w_{ij}x_j(t) + I_i(t)$, kde $I_i(t) = in_{ext}(t) + \theta_i$. Ako vidíme, tak adaptačné pravidlo je Hebbovho typu (1.6). Zosilnenie nastáva, ak sú znamienka synaptických váh rovnaké, inak nastáva zoslabenie.

Problémom pri týchto sietiach je však malé množstvo uchovateľných vzoriek a to približne 0.144 N, pretože pri hľadaní globálnej stability musí byť rozdiel energie záporný a musí vykazovať klesajúci charakter pri zvolení *Ljapunovej funkcie pre energie*.

2.1.2 SPÄTNÉ ŠÍRENIE CHYBY (ERROR BACKPROPAGATION) V REKURENTNÝCH NEURÓNOVÝCH SIETIACH (RES)

Z dôvodu rekurntných synapsií bude pre nás veľmi dôležitá dynamika činnosti RE (rekurentných) sietí. Teda vždy sa najskôr budú musieť vytvoriť podmienky pre globálnu stabilitu a chyba sa bude šíriť až po tomto kroku, aby bola sieť vždy stabilná. Takýto postup je dôležitý preto, aby nevznikli nekontrolovateľné procesy. Teda sieť musí byť stabilná ako pri šírení signálu dopredu, tak aj pri spätnom šírení.

Dynamika systému bude funkcia D(x(t)), kde x(t) je stav neurónu v čase t.

$$D\big(x(t)\big) = \frac{\boldsymbol{\partial} x(t)}{\partial t}$$

Riešenie budeme hľadať v pravidle zmeny synaptických váh tak, aby sme minimalizovali chybu a aby sme mali podmienky na dosiahnutie stavu stability v bode $x^*(t)$, keď je $D(x^*(t)) = 0$.

Jedným z východisiek je použitie *Ressistance-Capacity* modelu neurónu. Na tento model sa dá pozerať aj ako na elektrický obvod, ktorý nám diania v modeli jednoduchšie objasňuje (Obrázok 11). Na danom obrázku sú potom podľa knihy *Neurónové siete Inžiniersky prístup (2. diel)[3]*:

- synaptické váhy w_{ij} vodivosťou medzi neurónmi i a j
- $x_i(t)$ priemerný potenciál neurónov závislý na neurónových vstupoch a výstupoch
- R_i odpor neurónovej membrány neurónu i
- C_i kapacita neurónovej membrány neurónu i
- $f(in_i(t))$ prahová funkcia neurónu i.

Po úpravách získame rovnicu vstupu do neurónu:

$$in_i(t) = \sum_j w_{ij}\, ou_j(t) + I_i,$$

kde I_i je vstup do neurónu z externého sveta (θ_i, samotný vstup). Aby došlo ku globálnej stabilite siete je nutné vypočítať bod stability $x^*(t)$ z rovnice determinujúcej podmienky rovnováhy v neuróne (teda $D(x_i(t)) = 0$) :

$$\tau \frac{\partial x_i}{\partial t} = -x_i(t) + \sum_j w_{ij}\, x_j(t) + I_i,$$

kde τ je funkcia R,C. Potom chybový rozdiel výstupnej funkcie $x_i(t) = ou_i(t)$ bude:

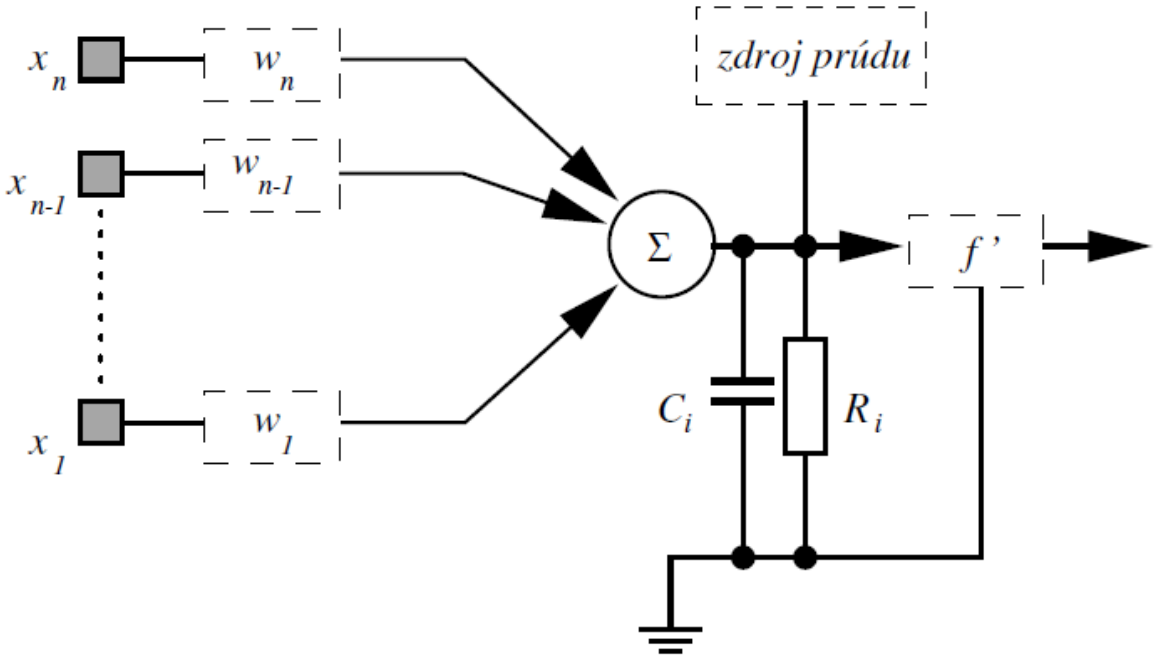
$$e_i(t) = \begin{cases} \big(ev_i(t) - x_i^*(t)\big) & pre\ vstupné\ neuróny \\ 0 & inak \end{cases}$$

kde $x_i^*(t)$ je stav výstupného neurónu *i* pri dosiahnutí rovnovážneho stavu a $ev_i(t)$ je očakávaná hodnota neurónu v zmysle kontrolovaného učenia. Chybová funkcia potom bude mať tvar:

$$J(t) = \frac{1}{2}\sum_k e_k(t)^2,$$

kde *k* sa mení od 1 po počet neurónov vo výstupnej vrstve. Všeobecný výraz pre adaptačné pravidlo potom bude:

$$\Delta w_{rs} = -\gamma \frac{\partial J}{\partial w_{rs}}.$$



Obrázok 11: *Ressistance-Capacity model neurónu* [3]

2.2 NEKONTROLOVANÉ UČENIE

Nekontrolované učenie je charakterizované metódami zhlukujúcimi dáta („*Adaptive Ressonance Theory*“, alebo „*Cluster discovery networks*“). Pri týchto metódach ale existuje problém neexistencie dôkazu globálnej stability, keďže prístupy boli ovplyvnené konkurenčným učením (východiskom môže byť postupná redukcia učiaceho parametra na 0, kedy nastáva globálna stabilita avšak na úkor adaptivity (prispôsobovaniu) k novým typom dát).

Jedným z cieľov ART je ponúknuť východisko pri probléme adaptívneho učenia sa. Keby sme použili napríklad učiaci algoritmus spätného šírenia chyby (*error backpropagation*) a doprednú sieť (*feed-forward*), tak by sme zistili, že pre každé naučenie sa nového prvku je nutné pamätať si aj všetky predchádzajúce prvky a sieť je nutné znova trénovať. Takže pri nutnosti prispôsobovania sa zmenám z vonkajšieho prostredia nie je možné tento prístup použiť a je potrebné siahnuť po prístupoch ako je ART, ktoré dosahujú *stability* aj *plasticity* systému.

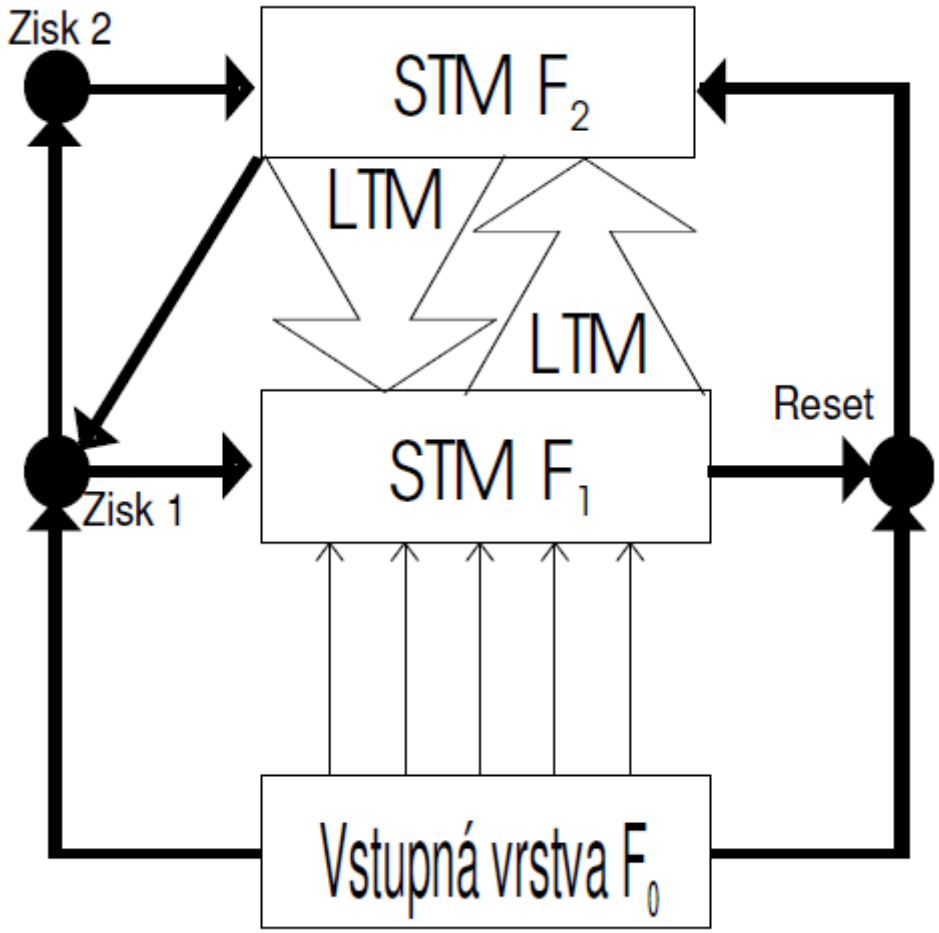
Pretože sa celý problém a metódy zaoberajú stabilitou a plasticitou, tak ako píše *Peter Sinčák* a *Gabriela Andrejková* [3] tu prikladám ich definíciu:

„Plasticita: Systém v priebehu učenia generuje rozpoznávacie kódy ako odozvu na postupnosť vstupných vzoriek z okolitého prostredia. Pritom sa postupne vytvárajú takzvané vzorky kritických čŕt, čo znamená, že každý rozpoznávací kód si ponechá len tie črty, ktoré ho odlišujú od kódov ostatných tried.

Stabilita: V priebehu učenia sa vytvárajú stabilné stavy, ktoré zabezpečujú priamu odozvu na už známu vzorku, pričom sa vnútorne rozhoduje o tom, či sa pre prezentovanú vzorku vytvorí nová kategória, alebo či bude iba upravený kód niektorej už existujúcej kategórie.“

ART funguje na princípe dosiahnutia rezonancie medzi dvomi vrstvami neurónov kde sa výstupné dáta presúvajú z jednej vrstvy na druhú. Rezonancia nastáva ak sieť už daný, alebo podobný vstup predtým spracovávala, alebo pokiaľ je vstup odlišný od všetkých ostatných. Vtedy sa začne zisťovať podobnosť už naučených vzoriek s danou prahovou hodnotou, ktorej vzorky musia vyhovieť. Pokiaľ rezonancia nenastáva, lebo daná vzorka je odlišná od všetkých predošlých a nespĺňa daný prah, tak sa vytvorí nová skupina rovnaká ako je vzorka, aby mohla byť daná vzorka zaradená a mohla teda nastať rezonancia.

Ďalšími príkladmi sú napríklad siete ART1 (Obrázok 12: *Architektúra neurónovej siete ART1*), ART-MAP a iné ich modifikácie.



Obrázok 12: *Architektúra neurónovej siete ART1* [3]

3 KOMPRESIA DÁT

Kompresia dát je zakódovanie dát (súboru) tak, aby ich celková veľkosť bola menšia, než veľkosť daných vstupných dát (súboru). Rozlišujeme dva typy kompresie: *stratovú* a *bezstratovú.* My sa budeme zaoberať *unikátne dekódovateľnými kompresiami* (teda nebude možmé ich dekódovať s rôznymi výsledkami). Iba takéto kompresie sú použiteľné pre moju prácu, pretože je nutné presné znovuzostavenie zakódovaných dát a nie je možné zostaviť sémantický slovník (v prípade *neunikátnej kompresie*) , ktorý by dokázal dekódovať všeobecné dáta zasielané a získavané z môjho programu.

Stratová kompresia je zmenšenie objemu dát s vedomím, že určité dátové reťazce sa stratia, alebo nahradia a výsledný skomprimovaný súbor už nebude možné obnoviť do stavu pred touto kompresiou. Príkladmi takýchto kompresií sú napríklad formáty:

.wav → .mp3,

.bmp → .JPEG.

Bezstratová kompresia zmenší objem dát bez toho, aby sa dáta stratili, alebo zmenili. Takáto konverzia je vždy vratná. Teda po dekompresii sa získa pôvodný súbor bez akejkoľvek zmeny. Napríklad formáty ako *.ZIP*, alebo *.7z*, či *.tar*.

3.1 N-GRAM

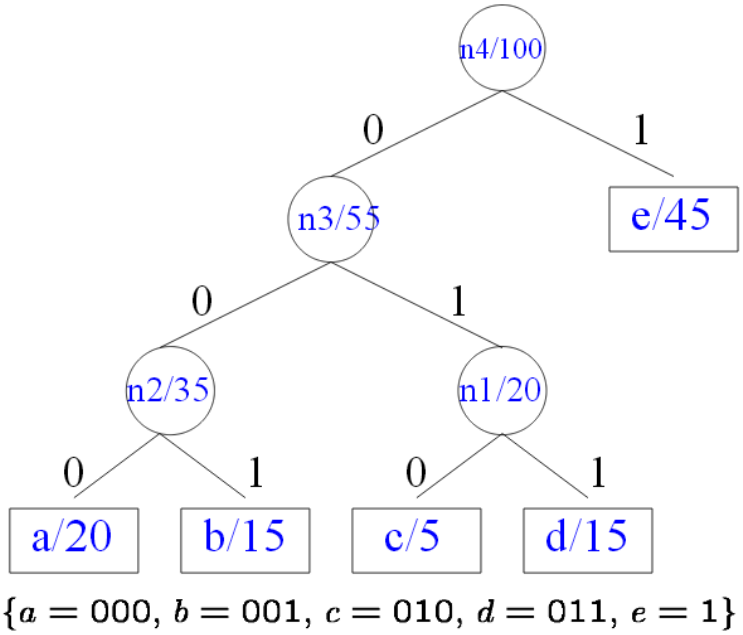
Ako sa píše na stránke *NgramJ* [16], ngram je väčšinou kratšia sekvencia atomických hodnôt (byty, znaky, slová, ...). Ngramový profil je (štatisticky) distribúcia ngramov. Teda ako často sa určitý ngram vyskytuje v danej sekvencii.

Vyhľadávanie týchto sekvencií v texte (v bitovom poli, ...) prebieha nasledujúcim spôsobom:

1.) Pre každú zložku (vlastnosť, alebo istá v konkrétnom ponímaní zmysluplná časť textu) sa vytvorí referenčný profil z danej textovej vzorky.
2.) Pri texte s neznámou zložkou sa určí textový profil a otestuje sa, či novo určený textový profil je podobný s niektorým už existujúcim referenčným profilom.

Toto bol všeobecný približný postup, ukazujúci všeobecný n-gramový princíp. (V skutočnosti je to zložitejšie z hľadiska určenia čo je to podobnosť zložiek.)

3.2 HUFFMANOV KÓD



Obrázok 13: *Huffmanov kód. Ukážka princípu na ktorom pracuje.* [9]

Tak ako vo svojich prednáškach opisuje profesor *Dakai WU* [9], Huffmanov kód je kombináciou binárneho stromu a prefixového kódu. Základnú myšlienku tvorí ekvivalencia medzi listami stromu a jednotlivými skupinami, ktoré sa snažíme zakódovať. Taktiež veľmi dôležitou skutočnosťou je rovnaká dĺžka kódovanej skupiny s hĺbkou danej skupiny v binárnom strome.

Pokiaľ označíme pojem *vstupné dátové skupiny* pojmom *abeceda* a jednotlivé *dátové skupiny* pojmom *písmeno*, potom môžeme algoritmus Huffmanovho kódu zapísať následovne:

1. Vyberieme dve písmená *x,y* z abecedy *A* s najnižšou frekvenciou výskytu v danom „texte“ (dátovom vstupe) a vytvoríme podstrom, ktorý má tieto písmená ako listy. Koreň tohoto podstromu nazveme ako *z*.
2. Nastavíme frekvenciu výskytu $f(z) = f(x) + f(y)$.
   A vytvoríme novú abecedu $A' = A \cup \{z\} - \{x,y\}$
   Teda $|A'| = |A| - 1$.
3. Opakujeme daný postup s abecedou *A´*, pokiaľ nezostane abeceda s jediným symbolom.

Výsledný strom je Huffmanov kód.

3.3 ARITMETICKÉ KÓDOVANIE

Táto metóda je náročnejšia na výpočetný výkon, lebo pracuje s desatinnými číslami, ktoré sú zložitejšie vzhľadom na to, že sa v nich musí počítať mantisa a exponent. Taktiež je táto metóda

pomalšia v porovnaní s Huffmanovým kódovaním [4], avšak má výhodu v lepšej kompresii dát a v možnosti menenia pravdepodobností počas kódovania správ.

Základným princípom je rozdeľovanie. V aritmetickom kódovaní sa pohybujeme v rozsahu od 0 do 1. Tento rozsah stále rozdeľujeme. Postup delenia je následujúci.

1. Najskôr si rozdelíme náš rozsah (od 0.0 do 1.0) na toľko častí, koľko máme možných znakov na kódovanie, pričom všetkým priradíme rovnaký rozsah.
2. Ďalšie delenie bude v skupine podľa prečítaného znaku, ale teraz už aj v závislosti na pravdepodobnosti výskytu jednotlivých znakov. Tento bod sa bude opakovať, kým takto nezakódujeme všetky znaky.
   Túto metódu nazývame *Korfhage Arithmetic Coding Method.*

<table>
<tr><th>Initial:
Divide area into 5 (# of unique characters)</th><th>1st Iteration: After A
Divide previous area by 6 (# unique characters +1)</th><th>2nd Iteration: After AB
Divide previous area by 7 (# unique characters +2)</th><th>3rd Iteration: After ABA
Divide previous area by 8 (# unique characters +3)</th></tr>
<tr><td rowspan="12">A: 0, 0.2</td><td>AA: 0, 0.067</td><td></td><td></td></tr>
<tr><td rowspan="11">AB: 0.067, 0.1</td><td rowspan="6">ABA: 0.067, 0.076</td><td>ABAA: 0.067, 0.070375</td></tr>
<tr><td>ABAB: 0.070375, 0.072625</td></tr>
<tr><td rowspan="3">ABAC: 0.07265, 0.07375</td></tr>
<tr></tr>
<tr><td>ABAS: 0.07375, 0.074875</td></tr>
<tr><td>ABAU: 0.074875, 0.076</td></tr>
<tr><td>ABB: 0.076, 0.086</td><td></td></tr>
<tr><td>ABC: 0.086, 0.091</td><td></td></tr>
<tr><td>ABS: 0.091, 0.095</td><td></td></tr>
<tr><td>ABU: 0.095, 0.1</td><td></td></tr>
<tr><td>AC: 0.1, 0.133</td><td></td><td></td></tr>
<tr><td>AS: 0.133, 0.167</td><td></td><td></td></tr>
<tr><td>AU: 0.167, 0.2</td><td></td><td></td></tr>
<tr><td>B: 0.2, 0.4</td><td></td><td></td><td></td></tr>
<tr><td>C: 0.4, 0.6</td><td></td><td></td><td></td></tr>
<tr><td>S: 0.6, 0.8</td><td></td><td></td><td></td></tr>
<tr><td>U: 0.8, 1.0</td><td></td><td></td><td></td></tr>
</table>

Obrázok 14: *Ukážka aritmetického kódovania pre slovo abacus* (skrátené z pôvodnej stránky) [17]

4 KOMPRESIA POMOCOU NEURÓNOVÝCH SIETÍ

Neurónové siete prinášajú možnosť vylepšenia dátovej kompresie oproti momentálne používaným n-gramovým modelom. S algoritmom popísaným v texte *Fast Text Compression with Neural Networks* [12] je možné dosiahnuť rýchlosti komprimovania približne 10^4 znakov za sekundu.

Výhody sa dajú pozorovať už v možnosti rozpoznávacích úloh, pretože neurónová sieť je schopná naučiť sa súvislosti. Napríklad okrem jednoduchých výskytových skutočností, že skupina znakov „áno“ v slovenčine sa vyskytuje častejšie ako skupina znakov „jsj“, tak aj zložitejšie prepojenia v podobe častého výskytu slov blízko seba v texte ako „oheň“ a „horí“.

Prvým námetom na takéto komprimovanie bola práca *Schmidhubera a Heila* (1996), ktorí predstavili možnosť použitia neurónových sietí na predikciu nasledovanú aritmetickým kódovaním. Metóda bola odvodená z modelu PPM (prediction by partial match) navrhnutého *Bellom, Wittenom a Clearym.*

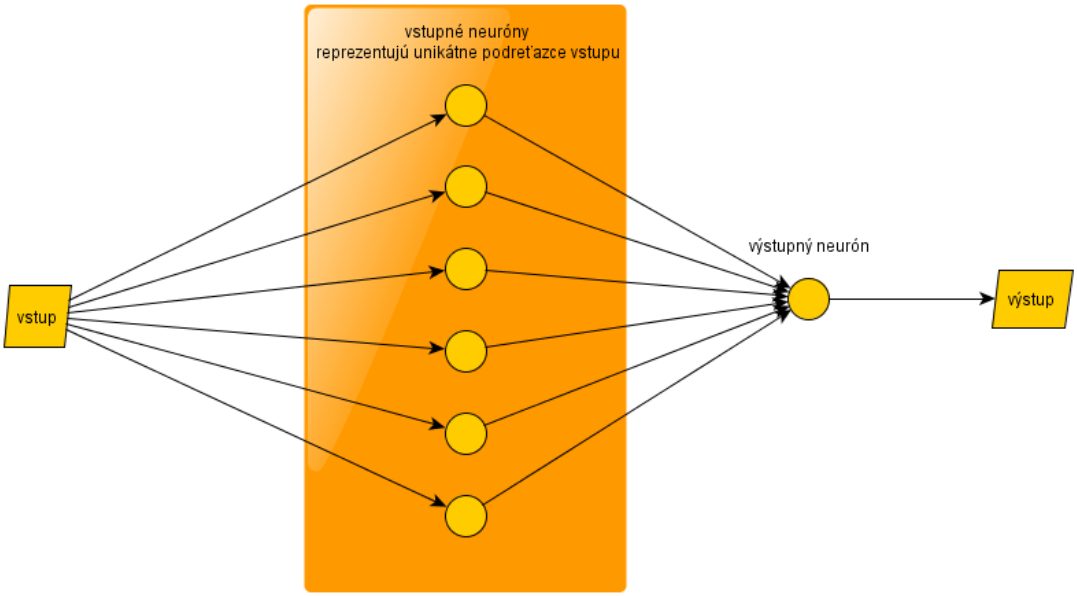
Ich predstavou bolo, že neurónová sieť po každom prečítanom bite predpovie aký bit bude následovať. Na túto predpoveď zareaguje aritmetický kóder, ktorý interval [0, 1] bude rozdeľovať (ako je popísaný v stati 0.) vzhľadom na optimálnosť aritmetického kódovania na jeden bit podľa Shannonovej limity.

Po tom, čo *Schmidhuber a Heil* nahradili PPM prediktor trojvrstvovou neurónovou sieťou a začali ho trénovať metódou backpropagation (viď 1.10) zistili, že daná metóda je príliš pomalá na uvedenie do praxe (niekoľko dní na superpočítači komprimovali dáta, ktoré bežné metódy bez neurónových sietí skomprimovali za dve sekundy).

5 PRAKTICKÁ ČASŤ

5.1 NÁVRH SPÔSOBU RIEŠENIA

Ako prvé si pri tejto problematike si treba uvedomiť, že nám stačí iba dvojvrstvá neurónová sieť. To vieme, lebo na vstupe v jednom kroku bude len jeden znak (jednotka alebo nula) a z neho nám stačí len jedna vrstva neurónov predstavujúca všetky doposiaľ (alebo do zvolenej veľkosti) zadané vstupy. Každý neurón v tejto vrstve bude obsahovať konkrétnu kombináciu, ktorá ho bude identifikovať a pravdepodobnosť vychádzajúcu z doposiaľ správnych a nesprávnych odhadov. Prvé výskyty zatiaľ v tejto úvahe ignorujeme, lebo by príliš zozložitili a znečitateľnili predkladanú myšlienku. Všetky tieto neuróny z prvej vrstvy sa prepoja synapsiami s jediným neurónom druhej vrstvy. Teda podľa synaptických váh potrebujeme určiť len jediný odhadovaný výstup.



Obrázok 15: *Topológia môjho riešenia*

Metóda, ktorú som si zvolil na úpravu váh je veľmi jenoduchá. Táto skutočnosť vychádza z faktu, že vieme v nasledujúcom kroku, po odhade následného znaku, povedať, či bol daný znak odhadnutý správne, alebo nie. Potom okamžite upravíme váhy učenia sa vzhľadom na vzniknutú chybu. Váhy môžeme upraviť pomocou sigmoidálnej aktivačnej funkcie.

5.1.1 AKTIVAČNÁ FUNKCIA

Ako zvoliť aktivačnú funkciu? Uvažujme. Vieme, že potrebujeme funkciu, ktorá nám umožní na začiatku učenia sa veľmi výrazne meniť synaptické váhy. Flexibilita na začiatku je potrebná, pretože nie je možné z prvých vstupov odhadovať nasledujúce znaky a tomuto odhadu dôverovať. Je teda nutné, aby sa tento odhad ľahko prispôsobil a menil odhad z nulovej na jednotkovú hodnotu.

Avšak pri častej chybovosti alebo opačne správnosti odhadu už je nutné, aby sa nám odhad príliš nemenil. Napríklad ak by sa nám daný neurón mýlil priemerne len jediný raz z dvadsiatich, tak je nežiaduce, aby sa na základe jedinej chyby stratila všetka dôvera danému neurónu. V takom prípade by došlo k odhadovaniu, ktoré by nám neposkytlo žiadne dôveryhodné odhady, lebo by bolo príliš ovplyvnené novými vstupnými reťazcami, ktoré by mali podobnú dôveru. Preto sa naša sieť musí utvrdzovať a viac dôverovať danému odhadu. Teda funkcia by mala pri opakovanom správnom predpovedaní pridávať stále viac dôvery, aby pri občasnom nesprávnom úsudku nestratila váhu. Táto skutočnosť platí aj obrátene pre často nesprávny odhad. V takom prípade by sa nemalo danému odhadu pridávať veľa dôvery, alebo by sa mala pridávať váha jeho nesprávnemu úsudku. To záleží na tom, či sa jedná o úsudok z ktorého sa dá zistiť správny odhad jeho invertovaním alebo či je odhad natoľko nepredvídateľný, že ho nie je možné dostatočne využiť.

Požiadavky na aktivačnú funkciu teda sú:

1. Musí byť flexibilná pri počiatočných zmenách
2. Musí sa utvrdzovať pri opakovane správnom, alebo nesprávnom odhade
3. Nesmie vkladať priveľkú dôveru (váhu) prípadom, ktoré sa chovajú nepredvídateľne.

Ja som najskôr uvažoval nad sigmoidálnou funkciou (Obrázok 4). Táto funkcia spĺňa všetky podmienky, ktoré som požadoval, pretože váhy na obrázku sa v y-ovej osi menia rýchlejšie v bode x = 0, ale čím viac sa blížia ku okrajom (hodnotám 0, alebo 1 na y-ovej osy), tým sa ich zmeny spomaľujú. To znamená, že pre synapsie na neurónoch, kde sa často odpovedalo správne, sa zvyšuje váha stále pomalšie.Teda sú síce veľmi ťažko zmeniteľné, ale zároveň dávame možnosť iným neurónom vo väčšom zastúpení zmeniť celkovú predpoveď a lepšie sa adaptovať na nové podnety.

Avšak nakoniec som sa rozhodol počítať nové váhy z chyby [6]. Teda prahy určujem tak, že si spočítam vzniknutú chybu po odhade v danom neuróne (neurón u mňa predstavuje jedinečný podreťazec vstupu). Podľa veľkosti chyby a učiacej konštanty si určím, koľko váhy sa pridá alebo odoberie danému neurónu a jeho schopnosti predpovedať nasledujúci bit. Tento prístup pokrýva všetky vyššie uvedené body - flexibilnosť pri počiatočných zmenách a dôvera sa nevkladá nepredvídateľným prípadom a zároveň sa utvrdzujú aj správne predpovedajúce neuróny. Naviac nám tento prístup ešte umožňuje rýchlejšie prípadne pomalšie meniť hodnoty váh, ktoré boli určené na začiatku. Zmena nastáva na základe chyby ktorú spravil, čo nám uľahčuje a zrýchľuje „zabúdanie“ dôvery (menenie dôvery na nedôveru). Ak sa napríklad v polovici kompresie zmenia hodnoty tak, že pôvodne nastavené hodnoty by boli pre zvyšok vstupných dát nepoužiteľné a bolo by nutné použiť iné reťazce na lepší odhad pravdepodobnosti ,tak sa váhy vďaka väčšej chybe zmenia oveľa rýchlejšie. Taktiež je tento prístup jednoduchší na výpočet. (vzorec a presný postup je popísaný v kapitole o implementácii 5.2)

5.1.2 UČENIE SA

V mojom programe som si zvolil online learning (1.9) pod dozorom (s učiteľom 1.5.1) ako typ učenia. Voľba bola v podstate veľmi jednoduchá, pretože:

1.) Vedel som zistiť aký bit nasleduje a vyhodnotiť, či som predpovedal správne, alebo nie. Teda som si mohol dovoliť zvoliť učenie s učiteľom.
2.) Ďalším faktorom bolo, že som potreboval sieť, čo sa bude rýchlo prispôsobovať a taktiež som išiel podľa myšlienky: *predpovedania len jedného následujúceho bitu*. Na tento druh obmieňania sa mi priam núkala možnosť použitia metódy online learning.

Ako som už spomínal, toto učenie mi vie zabezpečiť zmenu v každom kroku, čo je pre moju prácu veľmi výhodné aj z hľadiska schopnosti určovania pravdepodobnosti následujúceho bitu prakticky hneď po prečítaní dvoch bitov.

5.1.3 ENTRÓPIA

Hlavným výstupom môjho programu je entrópia. Táto hodnota nám udáva vypočíanie kompresného pomeru na základe chyby konečných odhadov v jednotlivých krokoch. Inak povedané, vieme vypočítať približný odhad v percentách podielu veľkostí vstupného a komprimovaného súboru.

Výpočet všeobecnej entrópie prebieha nasledujúcim spôsobom:

1.) Najskôr sa určia pravdepodobnosti jednotlivých javov tak, aby súčet všetkých týchto pravdepodobností bol rovný 1.
2.) Ďalej sa už len spočíta entropia podľa vzorca:

$$H = -\sum_{i=1}^{n} p_i * log_2(p_i),$$

kde p_i je pravdepodobnostné rozloženie ($\sum_{i=1}^{n} p_i = 1$), H je hľadaná entropia a n je počet zložiek pravdepodobnostného rozloženia.

5.2 IMPLEMENTÁCIA

Program začína postupným načítaním vstupu po bitoch, pričom spracovanie načítaných dát prebieha okamžite. Teda po každom načítanom bite sa pre nasledujúci bit vykonajú všetky operácie skôr ako sa začne načítavať ďalší bit. Takéto spracovanie je vhodné, pretože nie je závislé na veľkosti spracovávaných dát a nemusí si celý súbor ukladať do medzipamäte, čo by mohlo spôsobiť prílišné spomalenie výpočtov. Spomalenie síce nenastáva vzhľadom na medzipamäť, avšak program nie je optimalizovaný, a preto nie je možné porovnávať rýchlosť výpočtov vzhľadom na iné dostupné komprimátory. Z toho vyplýva zameranie na možnú kompresiu voči iným programom. Táto kompresia je určená vypočítanou entrópiou.

Požadovaný výstup z programu si určuje užívateľ pomocou číselných hodnôt v poli *output coments* (viď príloha ..\navodNaPouzitie\NavodNaPouzitie.docx).

Prvým krokom je cyklus načítania po bite. Pre každý bit, čo príde zo vstupu sa vypočíta pravdepodobnosť s akou bude za ním nasledovať bit s hodnotou logická 1. Táto pravdepodobnosť sa bude počítať na základe aritmetického priemeru z jednotlivých čiastočných výpočtov pravdepodobností, podreťazcov a dôvery (prahu na synapsii) pre každý z nich. Na ujasnenie prikladám vzorec:

$$P = \sum_{i=1}^{n} \frac{P_i * t_i}{n},$$

kde P je celková pravdepodobnosť, n je počet podreťazcov, z ktorých sa bude počítať odhad (nemusí byť použitý každý naučený podreťazec), P_i je konkrétna pravdepodobnosť podreťazca, t_i je váha na synapsii, $P_i * t_i$ v tomto prípade znamenajú určenú pravdepodobnosť s pridelenou dôverou (teda * v tomto prípade neoznačuje operáciu násobenia, ale operáciu pridelenia dôveryhodnosti jednotlivým pravdepodobnostiam).

Dôveru jednotlivým pravdepodobnostiam prideľujem nasledovne. Pravdepodobnosť započítam toľkokrát, koľko má jednotlivých atomických bodov dôvery (prahu). Takto to pridelím pre všetky pravdepodobnosti podreťazcov a nakoniec ich vydelím počtom všetkých použitých atomických bodov. Ide teda o jednoduchý aritmetický priemer, ktorý pre väčšiu názornosť môžme zapísať:

$$P = \frac{\sum_{i=1}^{n}(P_i * t_i)}{\sum_{i=1}^{n} t_i},$$

kde t_i je konkrétny počet bodov dôvery (prahu), P je celková pravdepodobnosť a P_i je pravdepodobnosť podreťazca.

Z tejto pravdepodobnosti sa nakoniec zistí predpoklad. Teda ak pravdepodobnosť, že nasledujúci bit má hodnotu logickej 1 je väčšia ako 0.5, tak sa bude odhadovať, že ďalší bit bude logická 1. Ak naopak bude hodnota menšia alebo rovná 0.5, nasledujúci prečítaný bit bude logická 0.

Hneď po určení pravdepodobností sa upravujú váhy na synapsiách. Nastavenie váh sa uskutočňuje pre všetky podreťazce vstupu po koniec súboru alebo po maximálnu dĺžku podreťazcov určenú užívateľom. Váha na synapsii, ďalej pomenovaná aj ako dôvera, sa zvyšuje pokiaľ daný podreťazec odhadoval nasledujúci znak správne, inak sa dôvera znižuje. Zvyšovanie a znižovanie dôvery sa vykonáva na základe určenej aktivačnej funkcie. V našom prípade to je táto funkcia:

$$t_i = t_{i-1} + c * E_i * P_i,$$

kde *i* je konkrétny podreťazec, *c* je učiaca konštanta (ako ľahko sa má sieť prispôsobovať, teda meniť dôveru v danom podreťazci), t_i je nová dôvera u tohoto podreťazca, t_{i-1} je predošlá dôvera tohoto podreťazca, P_i je pravdepodobnosť, že nasledujúci symbol bude logická 1 daného podreťazca a E_i je chyba pre tento podreťazec z odhadovanej a skutočnej hodnoty.

Chyba sa spočíta týmto spôsobom:

$$E_i = P_i * (1 - P_i) * (T - P_i),$$

kde T je hodnota bitu, ktorý skutočne prišiel.

Keďže by nebolo vhodné aby váha dosiahla nulovú alebo maximálnu hodnotu, tak je maximálna a minimálna hodnota pevne nastavená. Dôvodom tohoto nastavenia je skutočnosť vysvetlená na nasledujúcom príklade: Daný podreťazec na začiatku učenia sa často mýli. Hodnota jeho dôvery na synapsii dosiahla nulu. To znamená nemožnosť zmeny daných váh. Teda v našom prípade by sa daný reťazec nebral do úvahy a celková učiaca schopnosť by mohla byť ochromená bez možnosti ďalšej flexibility. Daný reťazec by sa už nikdy nebral do úvahy pri určovaní ďalšieho odhadu následujúceho znaku. To isté by platilo aj pre reťazec, ktorý by dosiahol maxima a do konca súboru by mal vždy pravdu.

Po nastavení váh podľa toho, či bol predchádzajúci odhad pravdivý alebo nie sa prestavia jednotlivé pravdepodobnosti podľa prídeného znaku. Prestavia sa len tie váhy, ktoré sú podreťazcami vstupu. V mojom spracovaní som entrópiu nepočítal z pravdepodobnosti či bol následujúci symbol 0 alebo 1, ale z pravdepodobnosti, či som uhádol správne alebo som neuhádol.

Tým chcem poukázať na nutnosť prepočítania odhadovaných pravdepodobností určujúcich výskyt jednotky alebo nuly na pravdepodobnosť určujúcu správny alebo nesprávny odhad. Tento kritický prepočet je nutné spraviť vzhľadom na to, že uvažovaná kompresia dát by nemala vychádzať z nahradenia reťazcov kratšími reťazcami ako pri bežných n-gramových kompresiách. Mala by byť založená na nahradzovaní kratšími reťazcami na základe správneho odhadu. Inak povedané, budeme zapisovať či sme sa mýlili alebo nie a kompresia by sa mohla generovať okamžite („online“). To umožňuje text skomprimovať v jednom cykle a oveľa rýchlejšie, než keby sme sa snažili najskôr nachádzať vhodné kombinácie a potom text znova prechádzali a komprimovali.

Ďalším krokom je výpočet samotnej entrópie. Pravdepodobnosť, z ktorej budeme vychádzať, spočítame z počtu správne predpovedaných pravdepodobností ku všetkým pravdepodobnostiam. Tieto súčty sa vyhodnocujú (pripočítavajú) v programe hneď ako sa vypočíta pravdepodobnosť nasledujúceho bitu. Po dokončení všetkých predpovedí a aj sčítavania správnych a všetkých predpovedí sa tieto súčty navzájom vydelia. Týmto spôsobom sme dostali pravdepodobnosť správneho odhadu. Na výpočet entrópie pre moje potreby používam vzorec:

$$H = -p * log_2(p).$$

kde p je pravdepodobnosť správnej predpovede.

5.3 VYHODNOTENIE VÝSTUPOV

Výsledky mojej práce sú dobre viditeľné na výstupoch. Najjednoduchšie je asi pozorovanie na chybovosti predpovedí, ktoré priamo súvisia s kompresiou. Testoval som 4 súbory s rôznymi veľkosťami, ktoré tu predkladám. (testovacie dáta: txt, bwp, lena, lenaNormal)

Súbory som testoval vo formáte .bmp a .png, pokiaľ sa jednalo o obrázky a .txt, pri textovom súbore. Textový súbor je generovaný pomocou stránky lorem ipsum (http://sk.lipsum.com/). Obrázky som prevzorkovával, aby mali približne požadovanú veľkosť.

5.3.1 .BMP FORMÁT S POČÍTANÍM PRAVDEPODOBNOSTÍ NA 4 BITOCH

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.bmp	bwp	2	4	10	8596	7963	16559
2			2	4	50	8519	8040	16559
3			2	4	100	8521	8038	16559
4			2	4	200	8512	8047	16559
5	.bmp	lena	2	4	10	9257	7814	17071
6			2	4	50	9242	7829	17071
7			2	4	100	9214	7857	17071
8			2	4	200	9152	7919	17071
9	.bmp	lenaNormal	2	4	10	9337	7734	17071
10			2	4	50	9334	7737	17071
11			2	4	100	9328	7743	17071
12			2	4	200	9324	7747	17071
13	.bmp	bwp	5	4	10	21478	20297	41775
14			5	4	50	21410	20365	41775
15			5	4	100	21410	20365	41775
16			5	4	200	21401	20374	41775
17	.bmp	lena	5	4	10	21820	19283	41103
18			5	4	50	21557	19546	41103
19			5	4	100	21489	19614	41103
20			5	4	200	21342	19761	41103
21	.bmp	lenaNormal	5	4	10	22418	18685	41103
22			5	4	50	22368	18735	41103
23			5	4	100	22361	18742	41103
24			5	4	200	22362	18741	41103
25	.bmp	bwp	10	4	10	43399	39976	83375
26			10	4	50	42786	40589	83375
27			10	4	100	42758	40617	83375
28			10	4	200	42662	40713	83375
29	.bmp	lena	10	4	10	44914	40477	85391
30			10	4	50	44471	40920	85391
31			10	4	100	44484	40907	85391
32			10	4	200	44295	41096	85391
33	.bmp	lenaNormal	10	4	10	46600	38791	85391
34			10	4	50	46627	38764	85391
35			10	4	100	46585	38806	85391
36			10	4	200	46530	38861	85391

Tabuľka 1: *Formát .bmp s pravdepodobnosťou počítanou na reťazci s dĺžkou 4 bity.*

Z Tabuľka 1 môžme pozorovať nasledujúce závery:

Pri maximálnej dĺžke reťazca obmedzenej na 4 bity nie je vidno súvislosti medzi nastavenou veľkosťou učiacej sa konštanty (learning constant) a správnosťou pri odhadoch nasledujúceho bitu

(chybovosť hádania). Môžeme si všimnúť, že pri hodnotách *learning constant* nastavených na hodnoty 50, 100, 200 nevidíme žiadne maximum, ktoré by bolo previazané s *chybovosťou hádania.* Hodnoty sú v niektorých prípadoch lepšie pre *learning constant* s hodnotou *100* ale v iných pre hodnotu *50* či *200*. Avšak nízka hodnota premennej *learning constant* má najvyššiu mieru správnych odhadov. Táto hodnota sa od ostatných hodnôt nelíši o viac ako približne 5% a to ani pri súboroch s veľkosťou 10 kB.

Taktiež sa nedá vnímať výraznejšia úspora, pretože hodnoty správnych a nesprávnych odhadov sú približne rovnako početné, čo znamená, že ušetrenie by bolo minimálne. Avšak už tu sa dá pozorovať, že sieť funguje správne a učí sa, pretože súbory s väčšou veľkosťou majú pomerovo viac dobrých odhadov ako súbory s menšou veľkosťou.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.bmp	bwp	2	8	10	8573	7986	16559
2			2	8	50	8431	8128	16559
3			2	8	100	8321	8238	16559
4			2	8	200	8323	8236	16559
5	.bmp	lena	2	8	10	9060	8011	17071
6			2	8	50	8621	8450	17071
7			2	8	100	8548	8523	17071
8			2	8	200	8606	8465	17071
9	.bmp	lenaNormal	2	8	10	9285	7786	17071
10			2	8	50	9254	7817	17071
11			2	8	100	9257	7814	17071
12			2	8	200	9273	7798	17071
13	.bmp	bwp	5	8	10	21517	20258	41775
14			5	8	50	21024	20751	41775
15			5	8	100	21014	20761	41775
16			5	8	200	20974	20801	41775
17	.bmp	lena	5	8	10	21707	19396	41103
18			5	8	50	19333	21770	41103
19			5	8	100	18399	22704	41103
20			5	8	200	18151	22952	41103
21	.bmp	lenaNormal	5	8	10	22642	18461	41103
22			5	8	50	22362	18741	41103
23			5	8	100	22282	18821	41103
24			5	8	200	22287	18816	41103
25	.bmp	bwp	10	8	10	43076	40299	83375
26			10	8	50	42048	41327	83375
27			10	8	100	41485	41890	83375
28			10	8	200	41442	41933	83375
29	.bmp	lena	10	8	10	45074	40317	85391
30			10	8	50	39732	45659	85391
31			10	8	100	38100	47291	85391
32			10	8	200	38171	47220	85391
33	.bmp	lenaNormal	10	8	10	47544	37847	85391
34			10	8	50	47073	38318	85391
35			10	8	100	46964	38427	85391
36			10	8	200	46928	38463	85391

Tabuľka 2: *Formát .bmp s pravdepodobnosťou počítanou na reťazci s dĺžkou 8 bitov.*

Z Tabuľka 2 môžeme pozorovať nasledujúce závery:

Pri maximálnej dĺžke reťazca obmedzenej na 8 bitov sa stále neprejavujú výraznejšie pomery ušetrenia, ba dokonca v riadkoch 18, 19 a 20 môžeme pozorovať zhoršenie. Avšak pri súboroch o veľkosti 10 kB už rozdiel medzi správnymi a nesprávnymi predpoveďami dosahuje 22 % (riadok 33). Opäť je najväčší rozdiel pri nastavení konštanty učenia sa na 10 z daných otestovaných hodnôt.

formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
					správne	nesprávne	celkovo[b]
.bmp	bwp	2	9	10	8658	7901	16559
		2	9	50	8642	7917	16559
		2	9	100	8566	7993	16559
		2	9	200	8553	8006	16559
.bmp	lena	2	9	10	10058	7013	17071
		2	9	50	9864	7207	17071
		2	9	100	9258	7813	17071
		2	9	200	9434	7637	17071
.bmp	lenaNormal	2	9	10	9347	7724	17071
		2	9	50	9390	7681	17071
		2	9	100	9312	7759	17071
		2	9	200	9310	7761	17071
.bmp	bwp	5	9	10	22182	19593	41775
		5	9	50	21753	20022	41775
		5	9	100	21257	20518	41775
		5	9	200	21169	20606	41775
.bmp	lena	5	9	10	24617	16486	41103
		5	9	50	24108	16995	41103
		5	9	100	22790	18313	41103
		5	9	200	23075	18028	41103
.bmp	lenaNormal	5	9	10	22798	18305	41103
		5	9	50	22749	18354	41103
		5	9	100	22570	18533	41103
		5	9	200	22542	18561	41103
.bmp	bwp	10	9	10	44915	38460	83375
		10	9	50	44285	39090	83375
		10	9	100	43503	39872	83375
		10	9	200	43245	40130	83375
.bmp	lena	10	9	10	52808	32583	85391
		10	9	50	51975	33416	85391
		10	9	100	48722	36669	85391
		10	9	200	49475	35916	85391
.bmp	lenaNormal	10	9	10	48016	37375	85391
		10	9	50	47690	37701	85391
		10	9	100	47333	38058	85391
		10	9	200	47271	38120	85391

Tabuľka 3: *Formát .bmp s pravdepodobnosťou počítanou na reťazci s dĺžkou 9 bitov.*

Pre Tabuľka 3 platí rovnaký záver ako pre Tabuľka 2 - až na vyčnievajúce zlepšenie pri čierno-bielom obrázku *„lena.bmp“*. Tento obrázok začal pri predpovedi na 9 bitov zlepšovať svoju predvídaciu schopnosť vďaka zápisu farieb. Pretože pre črierny pixel nám stačí vedieť len umiestnenie daného pixela a nepotrebujeme vedieť pomer farebných zložiek, čo nám ušetrí miesto. Neurónová sieť, ktorá má potom o jeden bit viac ako je zápis jedného pixelu potom „vidí“ aj následok predpovede. Teda vie po istej kombinácii ôsmych bitov (jedného pixelu) odhadnúť pozíciu ďalšieho

pixelu oveľa lepšie, ako kombinácia, v ktorej sa informácia neprelína s nasledujúcim zápisom pozície ďalšieho pixela.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.bmp	bwp	2	16	10	9315	7244	16559
2			2	16	50	9670	6889	16559
3			2	16	100	9709	6850	16559
4			2	16	200	9728	6831	16559
5	.bmp	lena	2	16	10	12973	4098	17071
6			2	16	50	13462	3609	17071
7			2	16	100	13601	3470	17071
8			2	16	200	13725	3346	17071
9	.bmp	lenaNormal	2	16	10	9514	7557	17071
10			2	16	50	9655	7416	17071
11			2	16	100	9672	7399	17071
12			2	16	200	9690	7381	17071
13	.bmp	bwp	5	16	10	24332	17443	41775
14			5	16	50	25198	16577	41775
15			5	16	100	25341	16434	41775
16			5	16	200	25486	16289	41775
17	.bmp	lena	5	16	10	30888	10215	41103
18			5	16	50	32106	8997	41103
19			5	16	100	32934	8169	41103
20			5	16	200	33663	7440	41103
21	.bmp	lenaNormal	5	16	10	23250	17853	41103
22			5	16	50	23732	17371	41103
23			5	16	100	23715	17388	41103
24			5	16	200	23731	17372	41103
25	.bmp	bwp	10	16	10	49549	33826	83375
26			10	16	50	51284	32091	83375
27			10	16	100	51634	31741	83375
28			10	16	200	51777	31598	83375
29	.bmp	lena	10	16	10	65533	19858	85391
30			10	16	50	69067	16324	85391
31			10	16	100	71103	14288	85391
32			10	16	200	72293	13098	85391
33	.bmp	lenaNormal	10	16	10	49447	35944	85391
34			10	16	50	50157	35234	85391
35			10	16	100	50141	35250	85391
36			10	16	200	50089	35302	85391

Tabuľka 4: *Formát .bmp s pravdepodobnosťou počítanou na reťazci s dĺžkou 16 bitov.*

Pri Tabuľka 4 sa dá odpozorovať obrátenie účinnosti učiacej konštanty (*„learning constant“*). Zatiaľ čo v predchádzajúcich tabuľkách dosahovala najvyššiu mieru správnych predpovedí, tu dosahuje mieru predpovedí najslabšiu.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.bmp	bwp	2	32	10	9357	7202	16559
2			2	32	50	9705	6854	16559
3			2	32	100	9747	6812	16559
4			2	32	200	9764	6795	16559
5	.bmp	lena	2	32	10	13103	3968	17071
6			2	32	50	13529	3542	17071
7			2	32	100	13653	3418	17071
8			2	32	200	13769	3302	17071
9	.bmp	lenaNormal	2	32	10	9553	7518	17071
10			2	32	50	9722	7349	17071
11			2	32	100	9747	7324	17071
12			2	32	200	9757	7314	17071
13	.bmp	bwp	5	32	10	24450	17325	41775
14			5	32	50	25244	16531	41775
15			5	32	100	25425	16350	41775
16			5	32	200	25578	16197	41775
17	.bmp	lena	5	32	10	31102	10001	41103
18			5	32	50	32304	8799	41103
19			5	32	100	33074	8029	41103
20			5	32	200	33745	7358	41103
21	.bmp	lenaNormal	5	32	10	23354	17749	41103
22			5	32	50	23865	17238	41103
23			5	32	100	23914	17189	41103
24			5	32	200	23917	17186	41103
25	.bmp	bwp	10	32	10	49899	33476	83375
26			10	32	50	51594	31781	83375
27			10	32	100	52068	31307	83375
28			10	32	200	52267	31108	83375
29	.bmp	lena	10	32	10	66169	19222	85391
30			10	32	50	69477	15914	85391
31			10	32	100	71193	14198	85391
32			10	32	200	72357	13034	85391
33	.bmp	lenaNormal	10	32	10	49839	35552	85391
34			10	32	50	50840	34551	85391
35			10	32	100	50955	34436	85391
36			10	32	200	50936	34455	85391

Tabuľka 5: *Formát .bmp s pravdepodobnosťou počítanou na reťazci s dĺžkou 32 bitov.*

V Tabuľka 5 najlepšie vidíme celkový rozdiel v predpovedaných hodnotách. Hlavne pri obrázku „*lena.bmp*“ je dobre badateľné, že odhad nasledujúcich predpovedí sa zlepšuje pri väčších súboroch. Ak porovnáme riadok 20 s riadkom 32, tak dostaneme približne takéto pomery predpovedí:

riadok 20: približne 6:1

riadok 33: približne 7:1

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.png	bwp	2	4	10	8240	7759	15999
2			2	4	50	8256	7743	15999
3			2	4	100	8250	7749	15999
4			2	4	200	8248	7751	15999
5	.png	lena	2	4	10	9710	9273	18983
6			2	4	50	9618	9365	18983
7			2	4	100	9552	9431	18983
8			2	4	200	9569	9414	18983
9	.png	lenaNormal	2	4	10	9786	9245	19031
10			2	4	50	9621	9410	19031
11			2	4	100	9622	9409	19031
12			2	4	200	9634	9397	19031
13	.png	bwp	5	4	10	21284	20675	41959
14			5	4	50	21112	20847	41959
15			5	4	100	21017	20942	41959
16			5	4	200	21024	20935	41959
17	.png	lena	5	4	10	24649	22502	47151
18			5	4	50	24553	22598	47151
19			5	4	100	24438	22713	47151
20			5	4	200	24415	22736	47151
21	.png	lenaNormal	5	4	10	22895	22360	45255
22			5	4	50	22762	22493	45255
23			5	4	100	22745	22510	45255
24			5	4	200	22772	22483	45255
25	.png	bwp	10	4	10	41625	39214	80839
26			10	4	50	40978	39861	80839
27			10	4	100	40907	39932	80839
28			10	4	200	40897	39942	80839
29	.png	lena	10	4	10	44531	42484	87015
30			10	4	50	44576	42439	87015
31			10	4	100	44333	42682	87015
32			10	4	200	43910	43105	87015
33	.png	lenaNormal	10	4	10	43151	41328	84479
34			10	4	50	42560	41919	84479
35			10	4	100	42615	41864	84479
36			10	4	200	42632	41847	84479

Tabuľka 6: *Formát .png s pravdepodobnosťou počítanou na reťazci s dĺžkou 4 bity.*

Pri formáte .png v Tabuľka 6 je vidieť podobnosť s Tabuľka 1. Ani tu sa nedá pozorovať zlepšenie. Súčty správne a nesprávne uhádnutých bitov sú približne rovnaké a pomerovo sa nelíšia ani pri iných veľkostiach súborov.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.png	bwp	2	8	10	8102	7897	15999
2			2	8	50	8207	7792	15999
3			2	8	100	8195	7804	15999
4			2	8	200	8174	7825	15999
5	.png	lena	2	8	10	9919	9064	18983
6			2	8	50	10006	8977	18983
7			2	8	100	9934	9049	18983
8			2	8	200	9943	9040	18983
9	.png	lenaNormal	2	8	10	9804	9227	19031
10			2	8	50	9794	9237	19031
11			2	8	100	9780	9251	19031
12			2	8	200	9724	9307	19031
13	.png	bwp	5	8	10	21481	20478	41959
14			5	8	50	21425	20534	41959
15			5	8	100	21286	20673	41959
16			5	8	200	21238	20721	41959
17	.png	lena	5	8	10	24891	22260	47151
18			5	8	50	25080	22071	47151
19			5	8	100	25094	22057	47151
20			5	8	200	25104	22047	47151
21	.png	lenaNormal	5	8	10	23285	21970	45255
22			5	8	50	23131	22124	45255
23			5	8	100	23062	22193	45255
24			5	8	200	22977	22278	45255
25	.png	bwp	10	8	10	41874	38965	80839
26			10	8	50	41497	39342	80839
27			10	8	100	41148	39691	80839
28			10	8	200	40937	39902	80839
29	.png	lena	10	8	10	45517	41498	87015
30			10	8	50	45215	41800	87015
31			10	8	100	45013	42002	87015
32			10	8	200	44917	42098	87015
33	.png	lenaNormal	10	8	10	43349	41130	84479
34			10	8	50	43283	41196	84479
35			10	8	100	43008	41471	84479
36			10	8	200	42967	41512	84479

Tabuľka 7: *Formát .png s pravdepodobnosťou počítanou na reťazci s dĺžkou 8 bitov.*

V Tabuľka 7 sa dá pozorovať podobnosť pomeru správnych predpovedí k nesprávnym s Tabuľka 6. Ak by sme si porovnali tieto výsledky na predpovediach založených na ôsmych bitoch s predpoveďami v odpovedajúcich si tabuľkách - Tabuľka 1 a Tabuľka 2 vo formáte bmp, tak dospejeme k zaujímavému zisteniu. Formát .png sa príliš nelíši v pomeroch či už predpoveď bola založená na štyroch alebo ôsmych bitoch. Pri formáte .bmp sa však tieto odhady zreteľne odlišujú.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.png	bwp	2	9	10	8157	7842	15999
2			2	9	50	8242	7757	15999
3			2	9	100	8225	7774	15999
4			2	9	200	8217	7782	15999
5	.png	lena	2	9	10	10012	8971	18983
6			2	9	50	10186	8797	18983
7			2	9	100	10187	8796	18983
8			2	9	200	10190	8793	18983
9	.png	lenaNormal	2	9	10	9816	9215	19031
10			2	9	50	9835	9196	19031
11			2	9	100	9832	9199	19031
12			2	9	200	9764	9267	19031
13	.png	bwp	5	9	10	21539	20420	41959
14			5	9	50	21413	20546	41959
15			5	9	100	21302	20657	41959
16			5	9	200	21226	20733	41959
17	.png	lena	5	9	10	25073	22078	47151
18			5	9	50	25306	21845	47151
19			5	9	100	25426	21725	47151
20			5	9	200	25453	21698	47151
21	.png	lenaNormal	5	9	10	23314	21941	45255
22			5	9	50	23243	22012	45255
23			5	9	100	23156	22099	45255
24			5	9	200	23141	22114	45255
25	.png	bwp	10	9	10	41884	38955	80839
26			10	9	50	41644	39195	80839
27			10	9	100	41468	39371	80839
28			10	9	200	41329	39510	80839
29	.png	lena	10	9	10	45551	41464	87015
30			10	9	50	45160	41855	87015
31			10	9	100	44938	42077	87015
32			10	9	200	44880	42135	87015
33	.png	lenaNormal	10	9	10	43704	40775	84479
34			10	9	50	43708	40771	84479
35			10	9	100	43497	40982	84479
36			10	9	200	43380	41099	84479

Tabuľka 8: *Formát .png s pravdepodobnosťou počítanou na reťazci s dĺžkou 9 bitov.*

Ako vidíme v Tabuľka 8, tak tu sa na rozdiel od Tabuľka 3 chybné odhady od správnych moc neodlišujú a to ani pri čierno-bielom obrázku *„lena.png“*.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.png	bwp	2	16	10	8187	7812	15999
2			2	16	50	8286	7713	15999
3			2	16	100	8294	7705	15999
4			2	16	200	8307	7692	15999
5	.png	lena	2	16	10	10129	8854	18983
6			2	16	50	10386	8597	18983
7			2	16	100	10486	8497	18983
8			2	16	200	10612	8371	18983
9	.png	lenaNormal	2	16	10	9829	9202	19031
10			2	16	50	9874	9157	19031
11			2	16	100	9924	9107	19031
12			2	16	200	9944	9087	19031
13	.png	bwp	5	16	10	21598	20361	41959
14			5	16	50	21797	20162	41959
15			5	16	100	21773	20186	41959
16			5	16	200	21869	20090	41959
17	.png	lena	5	16	10	24992	22159	47151
18			5	16	50	25166	21985	47151
19			5	16	100	25296	21855	47151
20			5	16	200	25322	21829	47151
21	.png	lenaNormal	5	16	10	23389	21866	45255
22			5	16	50	23572	21683	45255
23			5	16	100	23599	21656	45255
24			5	16	200	23602	21653	45255
25	.png	bwp	10	16	10	42041	38798	80839
26			10	16	50	42310	38529	80839
27			10	16	100	42341	38498	80839
28			10	16	200	42437	38402	80839
29	.png	lena	10	16	10	45768	41247	87015
30			10	16	50	45876	41139	87015
31			10	16	100	45881	41134	87015
32			10	16	200	45958	41057	87015
33	.png	lenaNormal	10	16	10	43625	40854	84479
34			10	16	50	43722	40757	84479
35			10	16	100	43769	40710	84479
36			10	16	200	43806	40673	84479

Tabuľka 9: *Formát .png s pravdepodobnosťou počítanou na reťazci s dĺžkou 16 bitov.*

V Tabuľka 9 pri formáte .png stále nie sú žiadne výraznejšie zmeny.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.png	bwp	2	32	10	8198	7801	15999
2			2	32	50	8304	7695	15999
3			2	32	100	8311	7688	15999
4			2	32	200	8322	7677	15999
5	.png	lena	2	32	10	10150	8833	18983
6			2	32	50	10409	8574	18983
7			2	32	100	10519	8464	18983
8			2	32	200	10635	8348	18983
9	.png	lenaNormal	2	32	10	9830	9201	19031
10			2	32	50	9878	9153	19031
11			2	32	100	9924	9107	19031
12			2	32	200	9945	9086	19031
13	.png	bwp	5	32	10	21611	20348	41959
14			5	32	50	21812	20147	41959
15			5	32	100	21775	20184	41959
16			5	32	200	21881	20078	41959
17	.png	lena	5	32	10	25000	22151	47151
18			5	32	50	25214	21937	47151
19			5	32	100	25338	21813	47151
20			5	32	200	25369	21782	47151
21	.png	lenaNormal	5	32	10	23385	21870	45255
22			5	32	50	23569	21686	45255
23			5	32	100	23596	21659	45255
24			5	32	200	23604	21651	45255
25	.png	bwp	10	32	10	41991	38848	80839
26			10	32	50	42327	38512	80839
27			10	32	100	42347	38492	80839
28			10	32	200	42407	38432	80839
29	.png	lena	10	32	10	45823	41192	87015
30			10	32	50	45913	41102	87015
31			10	32	100	45972	41043	87015
32			10	32	200	46041	40974	87015
33	.png	lenaNormal	10	32	10	43556	40923	84479
34			10	32	50	43628	40851	84479
35			10	32	100	43710	40769	84479
36			10	32	200	43732	40747	84479

Tabuľka 10: *Formát .png s pravdepodobnosťou počítanou na reťazci s dĺžkou 32 bitov.*

Ani Tabuľka 10 neprináša žiadny úspech.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.txt	txt	2	4	10	8851	7252	16103
2			2	4	50	8850	7253	16103
3			2	4	100	8866	7237	16103
4			2	4	200	8850	7253	16103
5	.txt	txt	5	4	10	22556	17739	40295
6			5	4	50	22489	17806	40295
7			5	4	100	22397	17898	40295
8			5	4	200	22392	17903	40295
9	.txt	txt	10	4	10	44866	35653	80519
10			10	4	50	44708	35811	80519
11			10	4	100	44650	35869	80519
12			10	4	200	44595	35924	80519

Tabuľka 11: *Formát .txt s pravdepodobnosťou počítanou na reťazci s dĺžkou 4 bity.*

Ako vidíme v Tabuľka 11, tak pri našom textovom dokumente sa prejavuje funkcia odhadovania už pri predpovedi na 4 bity. Predpoveď taktiež nie je závislá na veľkosti súboru, pretože jednotlivé pomery sú približne rovnaké. Taktiež sa zdá, že učiaca konštanta nemá veľký vplyv na tieto predpovede.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.txt	txt	2	8	10	10863	5240	16103
2			2	8	50	10809	5294	16103
3			2	8	100	10697	5406	16103
4			2	8	200	10575	5528	16103
5	.txt	txt	5	8	10	26993	13302	40295
6			5	8	50	27178	13117	40295
7			5	8	100	26845	13450	40295
8			5	8	200	26705	13590	40295
9	.txt	txt	10	8	10	54936	25583	80519
10			10	8	50	54464	26055	80519
11			10	8	100	53770	26749	80519
12			10	8	200	53479	27040	80519

Tabuľka 12: *Formát .txt s pravdepodobnosťou počítanou na reťazci s dĺžkou 8 bitov.*

Tabuľka 12 nám poskytuje už veľmi schopné predpovede a to iba na ôsmych bitoch. Avšak zmena konštanty učenia neprináša zatiaľ žiadne veľké rozdiely.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.txt	txt	2	9	10	11070	5033	16103
2			2	9	50	11198	4905	16103
3			2	9	100	11046	5057	16103
4			2	9	200	11016	5087	16103
5	.txt	txt	5	9	10	27640	12655	40295
6			5	9	50	28119	12176	40295
7			5	9	100	27538	12757	40295
8			5	9	200	27532	12763	40295
9	.txt	txt	10	9	10	56470	24049	80519
10			10	9	50	56613	23906	80519
11			10	9	100	55365	25154	80519
12			10	9	200	55559	24960	80519

Tabuľka 13: *Formát .txt s pravdepodobnosťou počítanou na reťazci s dĺžkou 9 bitov.*

Oproti Tabuľka 12 je možné v Tabuľka 13 pozorovať len veľmi drobné zmeny.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.txt	txt	2	16	10	11509	4594	16103
2			2	16	50	11771	4332	16103
3			2	16	100	11824	4279	16103
4			2	16	200	11866	4237	16103
5	.txt	txt	5	16	10	29182	11113	40295
6			5	16	50	30059	10236	40295
7			5	16	100	30247	10048	40295
8			5	16	200	30433	9862	40295
9	.txt	txt	10	16	10	59864	20655	80519
10			10	16	50	61061	19458	80519
11			10	16	100	61467	19052	80519
12			10	16	200	61941	18578	80519

Tabuľka 14: *Formát .txt s pravdepodobnosťou počítanou na reťazci s dĺžkou 16 bitov.*

Ako vidíme v Tabuľka 14, tak pomer odhadovania oproti Tabuľka 13 sa výrazne zlepšil. Taktiež môžeme pozorovať, že ako v odpovedajúcich tabuľkách v .bmp formáte sa začína prejavovať vlyv učiacej konštanty, ktorá je najhoršia pre hodnotu 10.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	chybovosť hádania		
						správne	nesprávne	celkovo[b]
1	.txt	txt	2	32	10	11748	4355	16103
2			2	32	50	12076	4027	16103
3			2	32	100	12192	3911	16103
4			2	32	200	12242	3861	16103
5	.txt	txt	5	32	10	30288	10007	40295
6			5	32	50	31326	8969	40295
7			5	32	100	31658	8637	40295
8			5	32	200	31919	8376	40295
9	.txt	txt	10	32	10	62823	17696	80519
10			10	32	50	64384	16135	80519
11			10	32	100	65180	15339	80519
12			10	32	200	65808	14711	80519

Tabuľka 15: *Formát .txt s pravdepodobnosťou počítanou na reťazci s dĺžkou 32 bitov.*

V Tabuľka 15 je veľmi dobre vidieť zlepšenie pomeru uhádnutých k neuhádnutým bitom pri zväčšovaní veľkosti súboru. Taktiež sa prejavil aj efekt učiacej konštanty, ktorý sa usporiadal vzostupne (od 10 do 200) a presne tak sa zlepšoval aj ich pomer uhádnutí.

	formát súboru	názov súboru	veľkosť súboru[kB]	combination length [bity]	learning constant	entrópia	približná veľkosť súboru [b]	pôvodná veľkosť v [b]
1	.bmp	lena	10	32	200	0,616405281	52635,46338	85392
2	.bmp	lena	10	16	200	0,61825554	52793,45879	85392
3	.bmp	lena	10	9	200	0,981735179	83831,34865	85392
4	.bmp	lena	10	8	200	0,991884089	84697,97424	85392
5	.bmp	lena	10	4	200	0,998987371	85304,53058	85392
6	.png	lena	10	32	200	0,997552606	86802,04005	87016
7	.png	lena	10	16	200	0,997710421	86815,77232	87016
8	.png	lena	10	9	200	0,999282019	86952,52487	87016
9	.png	lena	10	8	200	0,999242779	86949,11045	87016
10	.png	lena	10	4	200	0,999938262	87009,62785	87016
11	.txt	txt	10	32	200	0,685952946	55232,24525	80520
12	.txt	txt	10	16	200	0,779278476	62746,72365	80520
13	.txt	txt	10	9	200	0,893160699	71916,40631	80520
14	.txt	txt	10	8	200	0,92076379	74138,97965	80520
15	.txt	txt	10	4	200	0,991618366	79844,11923	80520

Tabuľka 16: *Vybrané hodnoty z tabuliek.*

Formát archívu	kompresná metóda	veľkosť slovníka [kB]	veľkosť slova	formát súboru	názov súboru	veľkosť pred komprimaciou [B]	veľkosť po komprimácii [B]	Veľkosť po v [b]
7z	LZMA	64	32	.bmp	lena	10 674	14 407	115256
7z	LZMA	64	32	.png	lena	10 877	11 086	88688
7z	LZMA	64	32	.txt	txt	10 065	3 261	26088
7z	PPmd	1024	32	.bmp	lena	10 674	4 511	36088
7z	PPmd	1024	32	.png	lena	10 877	11 184	89472
7z	PPmd	1024	32	.txt	txt	10 065	2 466	19728
7z	BZip2	100	-	.bmp	lena	10 674	4 481	35848
7z	BZip2	100	-	.png	lena	10 877	11 452	91616
7z	BZip2	100	-	.txt	txt	10 065	2 778	22224
BZip2	BZip2	100	-	.bmp	lena	10 674	4 373	34984
BZip2	BZip2	100	-	.png	lena	10 877	11 344	90752
BZip2	BZip2	100	-	.txt	txt	10 065	2 672	21376
Gzip	Deflate	32	32	.bmp	lena	10 674	5 067	40536
Gzip	Deflate	32	32	.png	lena	10 877	10 909	87272
Gzip	Deflate	32	32	.txt	txt	10 065	3 208	25664
Zip	Deflate	32	32	.bmp	lena	10 674	5 097	40776
Zip	Deflate	32	32	.png	lena	10 877	10 991	87928
Zip	Deflate	32	32	.txt	txt	10 065	3 494	27952
Zip	Deflate64	64	32	.bmp	lena	10 674	5 097	40776
Zip	Deflate64	64	32	.png	lena	10 877	10 991	87928
Zip	Deflate64	64	32	.txt	txt	10 065	3 494	27952
Zip	BZip2	100	-	.bmp	lena	10 674	4 487	35896
Zip	BZip2	100	-	.png	lena	10 877	10 991	87928
Zip	BZip2	100	-	.txt	txt	10 065	2 784	22272
Zip	LZMA	64	32	.bmp	lena	10 674	3 783	30264
Zip	LZMA	64	32	.png	lena	10 877	10 991	87928
Zip	LZMA	64	32	.txt	txt	10 065	3 438	27504

Tabuľka 17: *Iné kompresné metódy (program 7-Zip 4.65) pre vybrané hodnoty z tabuliek (testovacie príklady sú rovnaké ako v* Tabuľka 16: *Vybrané hodnoty z tabuliek).*

6 ZÁVER

Moja práca bola zameraná na preskúmanie možnosti komprimovania súborou pomocou neurónových sietí. V práci sa podarilo dokázať, že tento smer riešenia by mohol byť použiteľný na predom nekomprimované súbory. Túto skutočnosť dokazujú hodnoty v tabuľkách. Pri formáte .png, ktorý je už sám o sebe kódovaný (komprimovaný) vykazuje sieť veľmi slabé výsledky odhadovania. Avšak v konfrontácii s inými kompresnými metódami je viditeľné, že aj tieto kompresie majú problémy s komprimáciou súborov vo formáte .png. Pri formátoch .bmp a .txt sú výsledky porovnateľné s odlišnými kompresnými metódami.

Najviac ma prekvapil výsledok komprimovania pri obrázku *„lena.bmp“*. Zistil som pri ňom, že sieti sa výnimočne darí odhadovať čierno-biely obrázok. Taktiež som potvrdil domnienku o tom, že pri väčších súboroch sa sieť zdokonaľuje v odhadovaní.

Sieť sa začína použiteľne učiť približne pri odhadovaní s nastavenou dĺžkou maximálneho reťazca na deväť bitov. Moje tvrdenie platí hlavne pre opisovaný obrázok *„lena.bmp“*. Pri súbore *„txt.txt“* vidíme zlepšenie dokonca už na ôsmych bitoch. Pri oboch súboroch je učenie dobre viditeľné na 10 kB verziách.

Uvedomujem si však, že som testoval len veľmi malé súbory, ktorých veľkosť nepresiahla 10 kB. Táto skutočnosť je založená na časovej náročnosti, ktorú môj program má. Náročnosť vzniká z niekoľkých dôvodov. Prvým a asi najväčším je to, že veľkosť hodnôt pamätaných si programom rastie kvadraticky na každý bit maximálnej povolenej dĺžky neurónu (reťazca) vo väčších súboroch. Ďalšie opozdenia nastávajú v neoptimalizovaní napísaného kódu, ktorý nepoužíva bitové pole, ale dátový typ „string“ na uloženie do medzipamäte. Výrazne vylepšiť časovú zložitosť by sa dalo aj rozdelením výpočtov do viacerých vlákien a to prípadne ešte spracovať časť výpočtov na grafickej karte.

Práca by sa dala rozšíriť viacerými smermi. Mohli by sme napríklad posilniť niektoré druhy naučených reťazcov a rozdeliť ich podľa dátových formátov. Iným spôsobom zlepšenia je posilnenie niektorých dĺžok reťazcov. Zaujímavejšou možnosťou by mohol byť pokus o kombináciu sigmoidálnej funkcie s riešením použitým v mojom riešení. Pri takomto spojení by bolo zaujímavé pozorovať prenastavovanie premennej udávajúcej prikladanie váhy viac na počítanie pomocou sigmoidálnej funkcie s nízkym ovplyvnením chybového parametra alebo uberanie váhy. Výzvou by bola aj možnosť predpovedania naraz viacerých bitových reťazcov, avšak tu by som sa obával či by tieto predpovede pridali na účinnosti implementovaného prístupu.

Do budúcna by som rovnako navrhoval pripojenie komprimačného programu, ktorý by nepoužíval metódy ako sú Huffmanovo kódovanie alebo aritmetické kódovanie. Tieto druhy by som videl v predstavovanom návrhu len ako druhotné a inšpiratívne.

Zameral by som sa na myšlienku „online“ komprimácie podľa odhadu ďalšieho bitu s poznačovaním si počtu bitov za koľko dôjde k chybnej predpovedi. Inou možnosťou poznačovania by bolo rozdelenie zápisu chýb do skupín s rovnakou predom dohodnutou veľkosťou. V tomto prípade by sa buď do skupiny zapísal počet za koľko bitov dôjde k chybe alebo by sa zapísala hodnota

v koľkých nasledujúcich skupinkách nedošlo k chybe. Rozlišovanie medzi zápismi by sa mohlo určovať napríklad podľa prvého bitu skupiny.

Súčasný vývoj kompresie dát je podľa najnovších výskumov čoraz viac orientovaný smerom k využívaniu práve neurónových sietí.

LITERATÚRA

[1] MASTERS, T.: *Practical Neural Network Recipes in C++*. San Diego: ACADEMIC PRESS, INC., 1993. 0-12-479040-2.

[2] SINČÁK, P.; ANDREJKOVÁ, G.: *Neurónové siete Inžiniersky prístup (1.diel).* Košice: Technická univerzita v Košiciach, Košice: Univerzita P.J. Šafárika v Košiciach, 1996.

[3] SINČÁK, P.; ANDREJKOVÁ, G.: *Neurónové siete Inžiniersky prístup (2.diel).* Košice: Technická univerzita v Košiciach, Košice: Univerzita P.J. Šafárika v Košiciach, 2002.

[4] VERMA, B.; BLUMENSTEIN, M.; KULKARNI, S.: *A Neural Network based Technique for Data Compression.* In *Proceedings of the IASTED International Conference on Modelling and Simulation, MSO '97*. Singapore, 1997. s. 12-16.

[5] DEKEL, O.: *From Online to Batch Learning with Cutoff-Averaging* In *Proceedings of the NIPS Neural Information Processing Systems Foundation,* Vancouver, B.C., 2008.

[6] FUKUMIZU, K.: *Dynamics of Batch Learning in Multilayer Neural Networks.*Wako, Saitama 351-0198, 1998. Dostupné z: <http://www.ism.ac.jp/~fukumizu/research.html>

[7] LASKOWSKI, K.: *Hebbian Learning, Principal Component Analysis, and Independent Component Analysis.* In *Artificial Neural Networks,* Carnegie Mellon University, Pittsburgh, 2006.

[8] ČERŇANSKÝ, M.: *Materiály k prednáškam; Neurónové siete.* Bratislava: Slovenská technická univerzita v Bratislave, 1993-.

[9] WU, D.: *Lecture 15: Huffman Coding.* Hong Kong: The Hong Kong University of Science and Technology, 2005.

[10] LINSTER, CH.: *LECTURE 4: The hebbian learning rule,* Ithaca NY: Cornell University

[11] VONDRÁK, I.: *Neuronové sítě,* Ostrava: Vysoká škola báňská – Technická univerzita Ostrava, 1994.

[12] MAHONEY, M.V.: *Fast Text Compression with Neural Networks*. IN *Research Society Conference, 2000. AAAI FLAIRS.*, *Proceedings of the Thirteenth International Florida Artificial Intelligence*, Orlando, 2000, ISBN 1-57735-113-4.

[13] ANGUELOV, B.: *Basic Neural Network Tutorial : C++ Implementation and Source Code,* [online]. [cit. 2008-04-23]. < http://takinginitiative.net/2008/04/23/basic-neural-network-tutorial-c-implementation-and-source-code/>

[14] HECHT-NIELSEN, R.: *Theory of the Backpropagation Neural Network* IN *Neural Networks, 1989. IJCN., International Joint Conference,* Washington, DC., 1989, s. 593 – 605 vol.1.

[15] LEVERINGTON, D.: *A Basic Introduction to Feedforward Backpropagation Neural Networks,* [online]. [cit. 2009].

<http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html>

[16] SPIELECK, F.: *NgramJ,* [online]. [cit. 2007-03-19] Mar. 2007, version 1.0

<http://ngramj.sourceforge.net/principle.html>

[17] BURNS, B.; CHIANG, CH.; FONDY, J.; XU, M.: *Arithmetic Coding Tutorial* [online]. [cit. 2003] <http://www.sis.pitt.edu/~jfondy/IS_2140/Arithmetic.htm#Introduction>

PRÍLOHA - CD

Priložené CD obsahuje:

A) Bc.exe – bakalárska práca – program na OS Windows
../bakalarskaPracaProgram
B) BakalarskaPraca – text bakalárskej práce
../bakalarskaPracaText
C) Zdrojové súbory programu Bc.exe v jazyku C#
../zdrojoveSubory
D) NavodNaPouzitie.pdf – návod na použitie programu bc1.exe
../navodNaPouzitie/NavodNaPouzitie.docx
E) Testovacie dáta – vzorové súbory použité na testovanie programu
../testovacieData
F) Ukážky výstupu z programu
../ukazkyVystupu